DESIGN AND CHARACTERIZATION OF AN EMBEDDED AMPEROMETRIC ANALYZER FOR FIELD-PORTABLE ELECTROCHEMICAL APPLICATIONS

A Thesis

Presented to the Faculty of the Graduate School

of Cornell University

In Partial Fulfillment of the Requirements for the Degree of

Master of Science

by

Yinan Tang

January 2011

ABSTRACT

A novel miniaturized amperometric analyzer with enhanced performance was developed to meet the need of field-portable, wide adaptable, user friendly, and cost effective electrochemical detection in various application fields. Based on the new MSP430FG479 microcontroller, the embedded system was designed to either operate as a stand-alone unit with an on-board fool-proof user interface, or run on the user's PC for advanced data analysis.

Amperometric detection is a powerful experimental approach for the study of electro-active compounds with clinical, industrial, environmental and agricultural importance. Inside the amperometric analyzer, a mini-potentiostat is designed to interface with a wide spectrum of electrochemical sensors. Implemented with grounded-auxillary electrode configuration, the mini-potentiostat controls oxidation-reduction reactions of target analytes at the working electrode surface by accurately maintaining a preset potential between the working (WE) and the reference electrode (RE), while measuring the redox current response at the auxillary electrode (AE) using a low-side resistive sensing mechanism. In an unstirred solution, at steady state, the rate of electron transfer is directly proportional to the concentration of the reactant. Therefore, the steady state current is used as an accurate measure of the analyte concentration in the sample.

Constructed using off-the-shelf IC chips and powered by a single 3V lithium coin cell battery, the potentiostat circuitry inside the amperometric analyzer is designed to meet potential control and current detection requirements for most existing electrochemical sensors. Featured with sub-picoampere sensitive, wide span and high accuracy current sensing, the specially designed current sensing block not only covers general application requirements, but it can also serve as a powerful tool in research

efforts toward the next generation of ultra-sensitive detection systems in biomolecular sensing. In addition to the superior electrochemical related specifications, various low-power design techniques have been applied to ensure a prolonged battery life, therefore making the system an ideal platform for amperometric monitoring in field appropriate applications.

Confirmed with a systematic evaluation of the device prototype, the system features a 4-month battery life with a single 3V lithium battery (220mAh), accurate bidirectional redox potential control from -1.2V to +1.2V, high resolution current detection from ±5pA to ±1mA covered by 4 software selectable detection levels, improved stability unaffected by electrochemical sensor impedance, and high EMI immunity without external shielding.

Finally, electrochemical analyses were also conducted using the device prototype. For a constant potential amperometric experiment, the experimental results of the system were compared with a standard bench top electrochemical workstation (the BAS Epsilon). Excellent agreement was attained over the whole detection range, which demonstrates the excellent potential control and current sensing abilities of the amperometric analyzer.

BIOGRAPHICAL SKETCH

The author grew up in Hangzhou, China where she completed her elementary and secondary education. Achieved First Class Honours upon graduation, she obtained her Bachelor's degree in science with a major in biochemistry and minor in chemistry from the University of Hong Kong. She entered the Master of Engineering program in Cornell University's Biomedical Engineering Department in September 2008. Equipped herself with knowledge and experiences on embedded system design after successful completion of the one-year MEng coursework, she transferred to Master of Science program under the supervision of Professor Antje Baeumner and has since conducted interdisciplinary research into the development of a miniaturized potentiostat for field portable electrochemical applications.

To my parents.

ACKNOWLEDGMENTS

For the successful completion of this thesis work, I would like to express my great appreciation to my advisors, Professor Antje Baeumner, for introducing me to electrochemical sensing techniques and for helping me throughout my Master program. Thank you for your advice and encouragement during difficult times of this work, and thank you for giving me the freedom I needed to learn and grow in failure.

I am grateful to Professor Richard Durst, Amit Lal and Harold G. Craighead for their generous scientific support. I would like to thank all my teachers in Cornell University, as their guidance and experience has influenced my way of thinking as an engineer and a person. I am also thankful to Sylvia Kwakye, Pangshun Zhu and Han Wang for their technical support and useful suggestions.

Finally, I wish to express my love and gratitude to my parents for their unconditional love and all my friends for being a part of my life.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1 Introduction

1.1 Motivation and scope of the thesis

Amperometric detection has been traditionally carried out using stationary bench-top potentiostat instruments. These systems are designed to perform multiplexed electrochemical experiments with high accuracy and sensitivity, but they are also prohibitively expensive. With the rapid development of electrochemical transducers for various on-site monitoring applications, market is calling for an alternative solution for amperometric monitoring, which is small in size, light weight, easy to use, while in the meantime, can offer powerful monitoring capability.

A miniaturized electrochemical detection platform (miniEC) developed in our lab a few years ago is a good example for such a system[1]. Based on the 2-electrode potentiostat in miniEC, in this research project, an embedded amperometric analyzer containing a 3-electrode potentiostat is designed and implemented with improved performances in various aspects. Constructed using off-the-shelf ICs and powered by a single 3V lithium coin battery, the system can perform high-accuracy, ultra-sensitive, wide range amperometric detection with different electrochemical sensors. Moreover, design constrains such as bill of material, energy efficiency, and ease of use were also taken into consideration in the system design. This interdisciplinary research project requires deep understanding of electrochemical amperometric detection process as well as knowledge in electronics, specifically, mixed-signal embedded system design.

1.2 Thesis Organization

In later sections of chapter 1, the reader is first provided with a short general introduction of the amperometric detection principle. A review of current field-portable amperometric systems is then conducted to provide an overview of problems and opportunities that coexist with this topic. Finally, the specification of the target amperometric analyzer is presented to end the introduction chapter.

Chapter 2 and 3 of the thesis are coming from two papers which were summarized from this study. Chapter 2 focuses on the systematic design of the mixed-signal embedded system, with special emphasis on the low power, low noise design techniques. Chapter 3 goes into special design features of the potentiostat circuitry, which enable the system to conduct ultra-sensitive, wide-span potential control and current sensing with a 3V battery as the power supply. A systematic characterization of the device prototype was conducted, and test results are included in the result and discussion section of each chapter.

Chapter 4 reports additional important design aspects of the system, which are not included in previous chapters due to their limited spaces. The schematic and layout considerations are presented along with several significant parts of the software.

Finally, in chapter 5, work performed in this study is summarized. The performance and capabilities of the system are compared with its ancestor. And areas for potential future improvement are presented at the end to close the thesis.

1.3 Electrochemical amperometric detection principle

Amperometric detection is a powerful experimental approaches in electrochemistry[2]. It is used very effectively to conduct quantitative analyses of various kinds of inorganic, organic and biochemical species. With state-of-the-art electrochemical sensor, the limit of detection of amperometric detection can go down to femto-molar range[3].

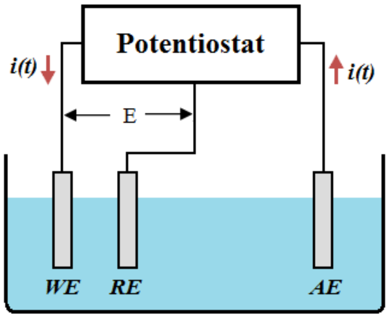


Figure 1.1: A schematic of a three-electrode amperometric detection setup. A redox potential E is applied between the working electrode (WE) and the reference electrode (RE), and the resulting electrochemical current i(t) is measured between the counter and working electrodes.

Figure 1.1 is a schematic showing the basic experimental setup. An instrument known as a potentiostat has control of the voltage across the working electrode (WE) and the reference electrode (RE). This potential drives the electrochemical reaction at the surface of the WE. The role of the RE is to establish a potential reference point, against which the WE potential can be assessed. The current produced from electrochemical reaction at the WE is balanced by a current flowing in the opposite direction at the auxiliary electrode (AE). This resultant current i(t), due to reduction or oxidation reaction, is a function of analyte concentration C and the imposed potential E, and is thus measured to quantify the species involved in the reaction.

To further illustrate the idea behind the three-electrode amperometric detection, a equivalent cell model is established[4] (Figure 1.2). $\triangle Ø_r$, $\triangle Ø_w$ and $\triangle Ø_a$ represent the double layer capacitances at the surface of RE, WE and AE respectively. Due to their high potential gradient (typically 10^6V/cm), all electrochemical reactions take place at double layer regions of the cell. Modeled with two resistors, the solution resistance is divided into two parts in a three-electrode system: a compensated resistance R_c and an uncompensated resistance R_u.

As indicated by the desired control point (DCP) in Figure X(a), the potential difference we wish to control is $\triangle Ø_w$. Since it is impossible to measure the electrolyte side of the double layer potential directly, a RE with well defined $\triangle Ø_r$ is used to setup the actual control point (ACP).

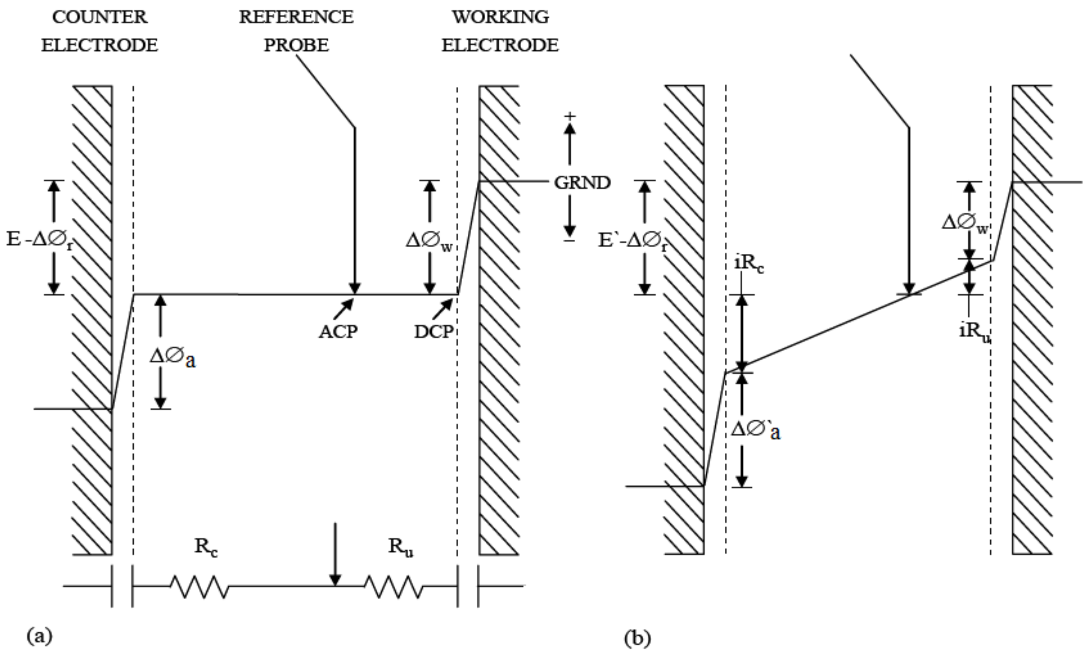


Figure 1.2: Schematic representation of potential gradients in a three-electrode cell: (a) i = 0; (b) i ≠ 0. Redraw from Kissinger 1984[4].

Under zero current condition (Figure 1.2(a)), there is no potential gradient across the bulk solution, so ACP equals to DCP. With a potential difference E applied between the WE and the RE, $\triangle Ø_w$ is actually controlled at E-$\triangle Ø_r$. When an elevated potential E' is applied to the system, electrochemical reaction at the WE's surface is initiated. The resultant current flow and the uncompensated resistance cause a small voltage error, iRu, cause the ACP to deviate from the DCP. The error term can be minimized by using a properly designed three-electrode cell, or by increasing the conductivity of electrolyte if allowed[4, 5].

For an electrode reaction, the relationship between the potential and the concentration of participating ions is governed by the following Nernst equation,

$$\mathrm{E} = \mathrm{E}_0 + 2.303\left(\frac{\mathrm{RT}}{\mathrm{nF}}\right)\log\frac{\mathrm{C_O}}{\mathrm{C_R}} \xRightarrow{\mathrm{T=298K}} \mathrm{E}_0 + \frac{0.0591\mathrm{V}}{\mathrm{n}}\log\frac{\mathrm{C_O}}{\mathrm{C_R}},$$

in which E_0 is the standard electrode potential[4]. When a potential step as shown in Figure 1.3(a) is applied to the WE[2], if this applied overpotential E_2 is well negative of the E_0, as the ratio of O and R at the surface of the WE is calculated to be less than 0.001 using the Nernst equation, it is safe to assume that the electroactive species O is consumed immediately at the WE's surface. So after time zero, a depletion region of O is formed adjacent to the electrode surface, which prompts a net diffusion of O from the bulk solution into the depleted region (Figure 1.3(b)). The Faradaic current is proportional to the concentration gradient of the species undergoing transformation. As the thickness of the depletion region increases with time, the slope of the concentration profile declines with time, and so does the current, until it reaches steady state which correlates to the steady state diffusion of the analyte to the electrode surface (Figure 1.3(c)).

Due to the correlation between the analyte concentration and the redox current, the resultant current profile is used as an accuracy and sensitivity measure of analyte concentration in various application fields. To control and monitor the dynamic amperometric process with high precision, the three-electrode potentiostat was first built by Hickling in 1942[6], and has experienced continuous innovation and improvement ever since.

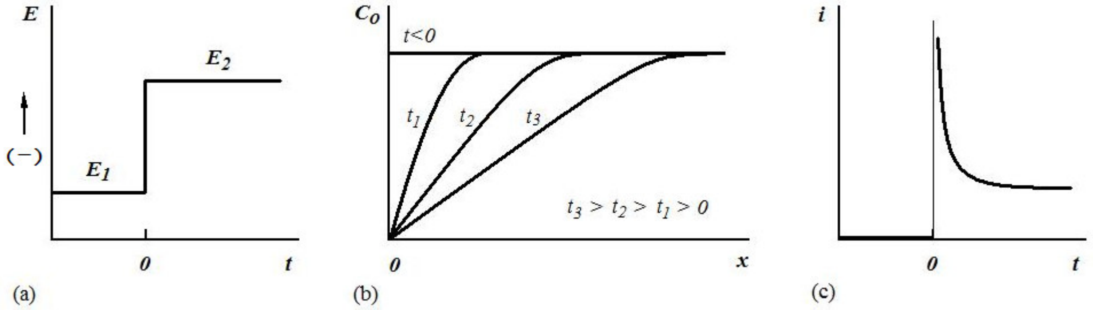


Figure 1.3: (a) Waveform of a potential excitation signal. (b) Concentration profile for various times into the experiment. (c) Current-time response signal[2].

1.4 A review of existing portable amperometric analyzer systems

Traditionally, amperometric detections were carried out using commercial bench-top electrochemical workstations, which are large in size and very expensive. They are designed for laboratory-based experiments and not intended for field-portable applications. In recent years, with the rapid development of electrochemical transducers for various on-site monitoring applications, several embedded amperometric detection systems were developed and presented in literature.

In Kwakye 2007, a miniaturized embedded system (miniEC) for electrochemical detection using a microfabricated electrode sensor array is introduced (Figure 1.4)[1]. Constructed using low-power high-performance off-the-shelf components and operated under event-driven software control, the system can run continuously for 2 weeks on a single AAA battery (1000mAh) [1]. The design specification of the 2-electrode potentiostat in miniEC is chosen to meet the detection need of a microfluidic biosensor[7]. A filtered pulse width modulation (PWM) signal is used to realize the excitation potential control, which ranges from 0 to 2500mV[1]. And the sensor current is amplified by a transimpedance amplifier circuit and measured using the MCU’s 12-bit ADC. The current sensing range covers 0-1250nA.

Though the resolution of the current detection was not reported in the paper, it is theoretically restricted by the resolution of the 12-bit ADC and thus cannot go beyond 0.6nA[1]. Though this 2-electrode potentiostat design serves well as a mini-amperometric transducer for a particular electrochemical microfluidic biosensor[7], however, if want to interface with other electrochemical sensors, a three-electrode design with enhanced potential control and current sensing ability is desirable.

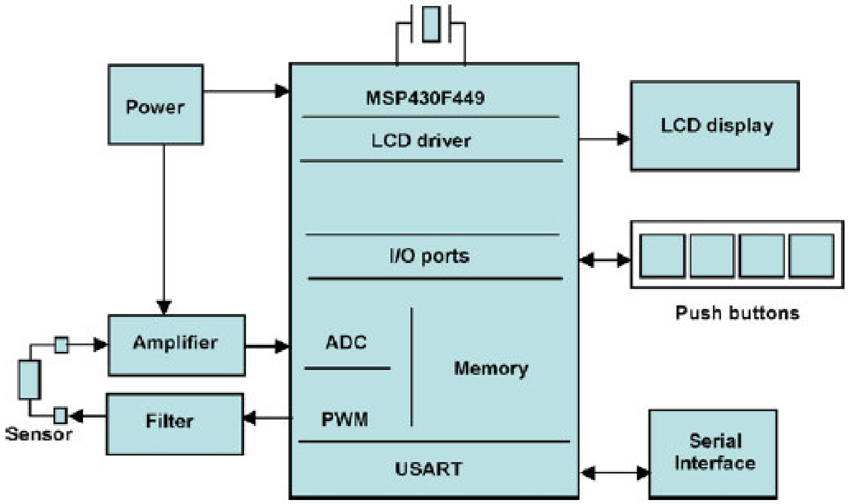


Figure 1.4: Block diagram of the MSP430 based miniEC system in Kwakye 2007 [1, 7].

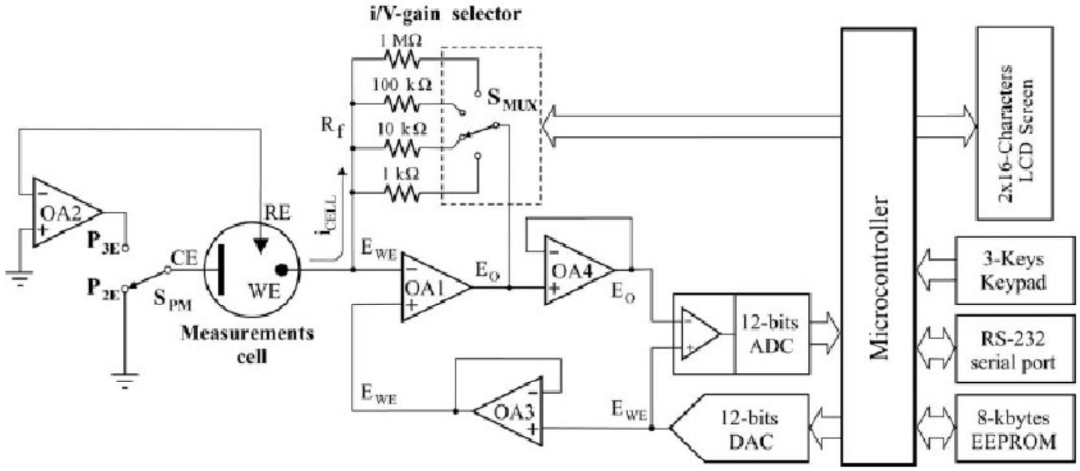


Figure 1.5: Block diagram of the amperometric analyzer in Avdikos 2005 [8].

In Avdikos 2005, a palm-sized microcontroller-operated analyzer for direct amperometric measurements is described (Figure 1.5) [8]. The low-power electronics in the system allow 8h of autonomous operation with a 9V battery (110mAh). The system is designed with selectable 2-electrode or 3-electrode potentiostat mode, and grounded-CE and grounded-RE configurations are used in the 2-electrode and 3-electrode mode respectively[8]. In both modes, the potential applied to the WE is developed at the output of a 16-bit DAC and is positive only. Proposed current sensing range (50nA-2mA) is covered by 4 different feedback resistors (1kΩ, 10kΩ, 100kΩ, 1MΩ) in the transimpedance amplifier. Automatic i/V gain selection is implemented with an analog multiplexer manipulated by the microcontroller. The measurement accuracy is reported to be ±0.05% of current range, which means only 10 useful bits of the 12-bit ADC can be achieved[8]. The major limitation of this design includes the positive only control potential (0-2V), and the relatively narrow current range. With its i/V gain selection design, a wider current range can be easily achieved with a more careful selection of feedback resistors. Moreover, it is possible to design a potentiostat with similar performance with a smaller, lighter and cheaper power source than a 9V battery.

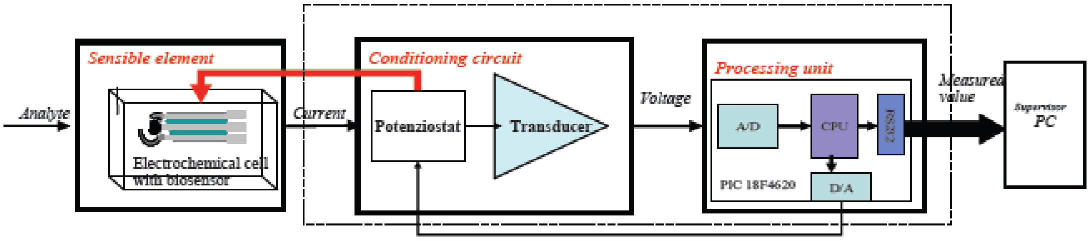


Figure1. Block diagram of the measurement unit

Figure 1.6: Block diagram of the measurement unit in Liguori 2009 [8].

A microcontroller based measurement unit for amperometric biosensor signal conditioning and processing is presented in Liguori 2009 (Figure 1.6) [9]. The potentiostat circuit is implemented with grounded-WE configuration with transimpedance amplifier for current sensing, and the hardware and software of the system is specially developed for automatic flow injection analysis. Similar to previous systems, only positive potential control (0-4V) is implemented in this system[9]. The performance of the current transducer was tested with several different biosensors, and the resultant limits of detection range from 5nA to 10nA[9]. A 2nA peak-to-peak noise is observed in all measurement results, which corresponds to 0.5% of the entire current detection range of the system (-200nA to 200nA). Therefore, an improved signal filtering strategy is desirable for enhanced detection sensitivity and accuracy.

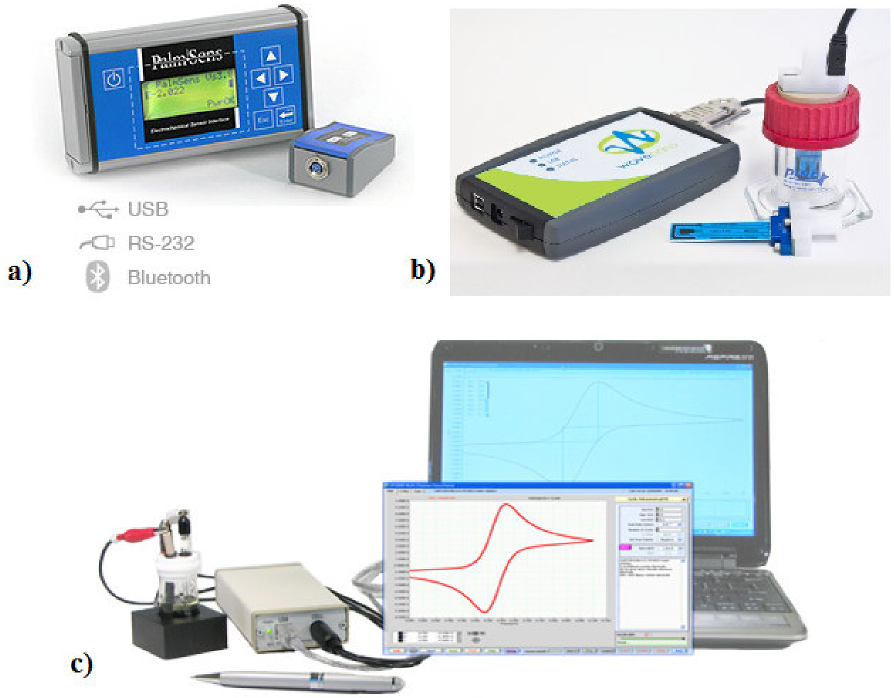


Figure 1.7: (a) The PalmSens handheld potentiostat/galvanostat. (b) The WaveNano USB Potentiostat. (c) The Digi-Ivy DY2100 mini-potentiostat.

With comprehensive discussion and comparison, Greef (1978) and Ahmadi (2009) reviewed various poteniostat topologies based on their potential control and current measurement techniques[10, 11]. Two main potential control configurations, specifically grounded-WE and grounded-AE, and two common current measurement methods, i.e. transimpedance amplifier and resistive current sensing, were widely used in previous potentiostat circuit design articles as well as commercially available potentiostat units[10, 11]. As each architecture has its own advantages and disadvantages, successful potentiostat designs are always based on specific application requirements and design constrains.

Table 1.1: Comparison of three commercial portable potentiostats.

	PalmSens	**WaveNano**	**Digi-Ivy DY2100**
Power source (operation time)	2 AA cells NiMH (8 hours)	5V Lithium Battery Pack (not available)	USB powered
Potential control range	± 2 V	± 4 V	± 2 V
Compliance voltage	± 8 V	>±10V	>± 2.2 V
Potential resolution	1 mV	n/a (16-bit DAC)	76µV (16-bit DAC)
Current range	±1 nA to ±10 mA (8 ranges)	±100 pA to ±1 mA (5 ranges)	±20nA to ±2mA (6 ranges)
Current resolution	0.1 % of current range 1 pA on lowest range	0.0034% of current range 3.4pA on lowest range	0.002% of current range 0.76pA on lowest range
Accuracy	≤ 0.2 % of current range at 100nA to 1 mA	±0.05% of range	n/a
Interfacing	RS232, USB, Bluetooth	USB	USB

Several commercial mini-potentiostat systems are newly developed to meet the increasing market demand in on-site monitoring field (Figure 1.7). Although design details of these commercial monitoring systems are not available to general public, a careful study of device specifications of three such systems, specifically PalmSens (Palm Instruments BV, Netherlands), WaveNano (Pine Research Instrumentation Inc., Raleigh, NC) and Digi-Ivy mini-potentiostat (Digi-Ivy, Inc., Austin, TX), is conducted to provide useful insights and guidelines in the development of our amperometric analyzer (Table 1.1).

As shown in Table 1.1, PamSens and WaveNano are truly portable, while Digi-Ivy mini-potentiostat can only be used together with a USB power source. In addition, the first two systems can function as a stand-alone devices as well as interface with PC or PDA through various communication links.

Potential control range is an important characteristic of a potentiostat, as it determines the application scope of the system. The standard redox potentials of the majority of electrochemically active species fall into the ± 2V range[12], and this most useful range is covered in all three designs (Table 1.1).

A trend in decreasing redox potential is observed in recently developed portable electrochemical sensors[14]. Specifically, in the latest generation of biosensors, small redox active molecules, such as ferrocene derivates, ferrocyanide, conducting organic salts and quinines, have been extensively used as mediators to shuttling the electrons between the analyte and the electrode, thus reducing the required applied potentials to under ± 1V [15]. Moreover, for amperometric experiment in aqueous phase, the redox potential for reliable detection is restricted by the stability of water[13], with upper and lower limitations ranging from -0.8V to +1.23V depending on the solution pH (Figure 1.8). In addition, when working in non-aqueous solution, electrochemists are required to clean their electrodes at potential that are between +1.1 to +1.2V, as well as -1.1 to -1.2V. Therefore, in order to interface with a wide spectrum of electrochemical sensors, there is a definite need for a potentiostat to have potential control ability reaching ±1.2V but not much further.

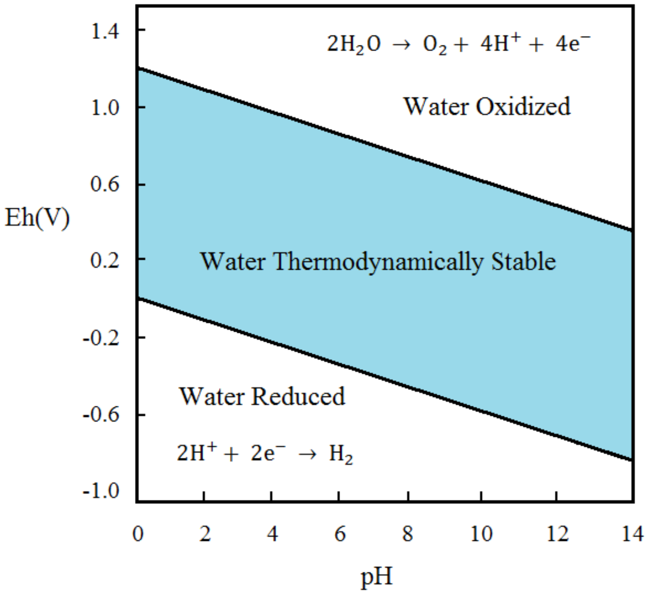


Figure 1.8: Eh-pH diagram showing the thermodynamically stable region of water. Eh is defined as the potential of a system relative to the potential of the standard hydrogen electrode (SHE) [13].

The potential control resolution of a potentiostat depends on both DAC resolution and noise level in the signal path. In all three systems, potential control function is realized through a16-bit DAC. In PalmSens, 1mV resolution is achieved, which means only 12 bits are useful in its 16-bit DAC due to noise issue. Although, the potential control resolution is not specified in WaveNano and Digi-Ivy systems, a similar resolution level can be expected. Therefore, an expensive 16-bit DAC is actually overkill considering the noise level, since a 12-bit DAC is adequate to achieve comparable potential control resolution.

The current detection range is another important characteristic of a potentiostat. As shown in Table 1.1, all three systems have similar current detection ability, with a lower limit of detection at the nA level and upper limits at the mA level. It reflects the current detection requirements of most existing field-portable applications. However, in many ultra-sensitive detection systems that are under development[16, 17], especially those for biomolecular sensing, a portable potentiostat with current detection ability down to pA range is highly desirable. Such a system will not only provide scientists with a powerful research tool, but will also, in the future, expand the pool of prospective applications and improve the user experience of future customers.

Current resolution is expressed as a percentage of the current range in the device specifications of all three systems. The 0.1%, 0.0034% and 0.002% current resolution corresponds to 10 bits, 15 bits and 15.5 bits ADC capability respectively. However, due to noise perturbation and system offset, not all current readings are accurate. So the accuracy of the current reading is a more stringent parameter than current resolution, as it reflects the ability of the instrument to measure current precisely and reliably. PalmSens has a current accuracy of 0.2%, which means, only 9 useful bits can be achieved from the 10-bit ADC resolution; while for WaveNano, ±0.05% accuracy corresponds to 10 useful bits out of a 15-bit ADC. High accuracy is hard to achieve at lower current range. Especially at picoampere level, current sensing using an embedded system reaches its theoretical limit, as bias current and leakage current of high quality IC chips are also at pA or sub-pA level, not to mention various noise sources exist internally and externally. Therefore, only when a good circuit design is combined with careful IC selection, good PCB layout practice, effective signal filtering and shielding techniques, current sensing in the pA range can be realized in a portable embedded system.

1.5 Target Device specification

The goal of this project is to design and develop a microcontroller-based amperometric analyzer for field-portable electrochemical sensing applications. After an in-depth understanding of amperometric detection principles as well as a careful study of existing systems, a list of desirable characteristics of an improved amperometric analyzer is presented below.

- Over 10-hour continuous operation with a single battery;
- Minimum power consumption when stand-by;
- Potential control range: at least -1.2V to 1.2V, with 1mV precision;
- Capable of sensing current from pA to mA level in several ranges;
- Current sensing resolution of at least 0.01% of current range;
- Current sensing accuracy of at least ±0.05% of current range;
- Can operate as a stand-alone device, while also support data transmission to PC for further analysis;
- Foolproof user interface for easy experiment control and result interpretation.

The detailed design, development and testing process of such a system is presented in later chapters of this thesis.

REFERENCES

1. Kwakye, S. and A. Baeumner, *An embedded system for portable electrochemical detection.* Sensors and Actuators B: Chemical, 2007. **123**(1): p. 336-343.

2. Bard, A.J. and L.R. Faulkner, *Electrochemical methods: fundamentals and applications*. 2nd ed. 2001, New York: John Wiley & Sons, Inc.

3. Xie, H., C. Zhang, and Z. Gao, *Amperometric Detection of Nucleic Acid at Femtomolar Levels with a Nucleic Acid/Electrochemical Activator Bilayer on Gold Electrode.* Analytical Chemistry, 2004. **76**(6): p. 1611-1617.

4. Kissinger, P.T. and W.R. Heineman, eds. *Laboratory Techniques in Electroanalytical Chemistry.* 1984, Marcel Dekker, Inc.: New York.

5. Scribner, T., ed. *The measurement and correction of electrolyte resistance in electrochemical tests*. 1990, American Society for Testing and Materials.

6. Roe, D.K., *Analytical electrochemistry: theory and instrumentation of dynamic techniques.* Analytical Chemistry, 1978. **50**(5): p. 9R-16R.

7. Kwakye, S., V.N. Goral, and A.J. Baeumner, *Electrochemical microfluidic biosensor for nucleic acid detection with integrated minipotentiostat.* Biosensors and Bioelectronics, 2006. **21**(12): p. 2217-2223.

8. Avdikos, E.M., M.I. Prodromidis, and C.E. Efstathiou, *Construction and analytical applications of a palm-sized microcontroller-based amperometric analyzer.* Sensors and Actuators B: Chemical, 2005. **107**(1): p. 372-378.

9. Liguori, C., V. Paciello, and A. Pietrosanto, *Microcontroller based measurement unit for biosensors*, in *6th International Multiconference on Systems, Signals and Devices, 2009*. 2009: Djerba.

10. Ahmadi, M.M. and G.A. Jullien, *Current-Mirror-Based Potentiostats for Three-Electrode Amperometric Electrochemical Sensors.* Circuits and Systems I: Regular Papers, IEEE Transactions on, 2009. **56**(7): p. 1339-1348.

11. Greef, R., *Instruments for use in electrode process research.* Journal of Physics E: Scientific Instruments, 1978. **11**(1): p. 1.

12. Kotz, J.C., P. Treichel, and J.R. Townsend, eds. *Chemistry and chemical reactivity*. Vol. 2. 2009, Thomson Brooks/Cole: Belmont.

13. DeLaune, R.D. and K.R. Reddy (2005) *Redox Potential*.

14. Pauliukaite, R., et al., *Phenazines and Polyphenazines in Electrochemical Sensors and Biosensors.* Analytical Letters, 2010. **43**(10): p. 1588 - 1608.

15. Freire, R.S., et al., *Direct electron transfer: an approach for electrochemical biosensors with higher selectivity and sensitivity.* Journal of the Brazilian Chemical Society, 2003. **14**: p. 230-243.

16. Hassibi, A. and T.H. Lee, *A Programmable 0.18-um CMOS electrochemical sensor microarray for biomolecular detection.* IEEE Sensors Journal, 2006. **6**(6): p. 1380-1389.

17. D'Orazio, P., *Electrochemical Sensors: A Review of Techniques and Applications in Point of Care Testing*. Point of Care, 2004. **3**(2): p. 49-59.

Chapter 2 Ultra-low Power Embedded System for Picoampere Amperometric Detection

2.1 Introduction

Electrochemistry is one of the oldest specialty areas of chemistry[1]. Since its origins in mid-19th century, the field of electrochemistry has evolved greatly with improved understanding of phenomena, continuous introduction of new methods and further development of experimental tools[1]. Nowadays, electrochemical detection has found extensive new applications for the study of electro-active compounds with clinical, industrial, environmental and agricultural importance.

Of all different electrochemical detection methods, amperometric detection has received remarkable attention, due to its high detection specificity and sensitivity, as well as its analytical simplicity and field portability[2, 3]. At the heart of any amperometric analyzer is its potentiostat, which controls oxidation-reduction (redox) reaction of target analytes at the electrode surface by accurately maintaining a preset potential between the working and a reference electrode, while measuring the redox current response between the working and auxillary electrode. The amount of resultant current is a function of the imposed voltage and the analyte concentration in the sample.

A huge array of sensing technologies has been developed utilizing amperometric detection principle. Numerous papers and reviews of amperometric chemical sensors and biosensors concerning clinical and therapeutic drug monitoring[2-4], food analysis[5, 6], environmental pollution monitoring[4, 7, 8], and industrial process monitoring[9] have been published in recent years. The rapid development of electrochemical transducers introduces the needs of a field portable, wide adaptable, user friendly, and cost effective amperometric detection platform for monitoring in various environmental conditions and geographic locations.

With home blood glucose monitors as a successful model for portable amperometric detection system, researchers[3, 10-12] have designed and fabricated microcontroller-based amperometric analyzers using off-the-shelf integrated circuits (IC). However, their scopes of application are restricted by limitations in design specifications, for instance, unipolar potential control, unidirectional current sensing, relative low sensitivity and small detection range.

Here we present a three-electrode embedded amperometric analyzer with enhanced potentiostat performance, including bipolar potential control, and ultra-sensitive bidirectional current sensing in the pA range. A major design challenge of field-appropriate devices is the strict requirements in size, weight and power. A focus on low-power design can help to solve all the portability requirements, as the system power consumption directly impacts the size and weight of the required batteries. Keeping portability consideration in mind, various low-power design techniques have been used here to keep the system running for a prolonged period of time with a single 3-V lithium coin cell battery. The robust design enables the system to be connected to different electrochemical sensor fronts in a variety of analytical applications. The paper introduces the systematic design of the embedded system with special focus on rationales behind various design decisions.

2.2 Hardware design of the amperometric analyzer

2.2.1. Overview of system design

The block diagram of the amperometric analyzer is presented in Figure 2.1. The embedded system consists of the following five modules: a core microcontroller unit (MCU), a USB communication interface, a power supply module, an analog potentiostat, and a user interface module[11].

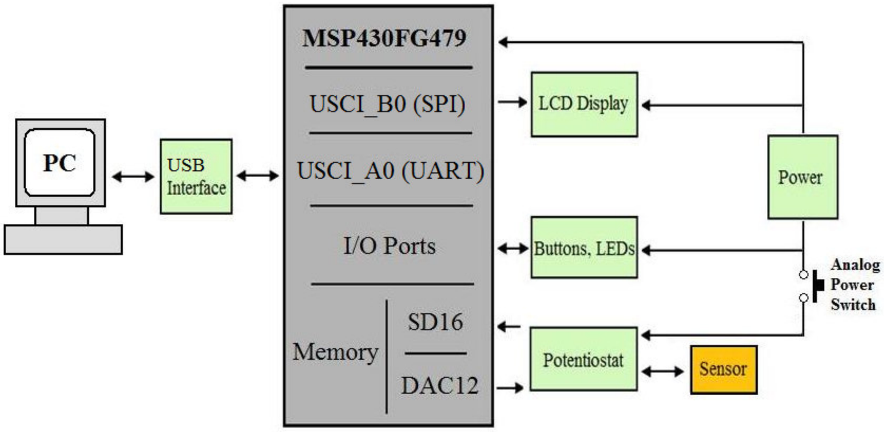


Figure 2.1: System block diagram of the amperometric analyzer.

To control and coordinate the whole system so that each module can perform its dedicated tasks, a 16-bit microcontroller MSP430FG479 (Texas Instruments, Dallas, Texas) is chosen as the heart of the analyzer. The architecture of MSP430FG479, combined with five low-power modes (LPM), is optimized to achieve extended battery life in portable measurement applications. With its internally configured high performance 16-bit sigma-delta analog-to-digital converter (ΣΔADC), dual 12-bit digital-to-analog converters (DAC), 48 I/O pins, supply voltage supervisor (SVS), two universal serial communication interfaces (USCI) and 60K flash memory, the MSP430FG479 microcontroller is specially designed for applications such as analog sensor systems and hand-held meters, and was therefore chosen for our system[13, 14].

2.2.2. Microcontroller Unit

In Kwakye (2006), four microcontrollers from different MCU family architectures, specifically, Atmega128L, MSP430F449, PIC18LF452 and MC9S12C64, were evaluated based on various design requirements of her portable electrochemical detection system. The MSP430F449 microcontroller was picked for its combination of desired system features[11]. The new MSP430FG479, also from MSP430 family of ultra-low-power microcontrollers, was first released in 2009. It belongs to the System on Chip (SoC) series in the MSP430 portfolio, because this device has integrated the entire signal chain on-chip[13]. By reducing the need for fewer external components, MSP430FG479 greatly simplifies the design of the system and reduces the material costs for the overall system.

2.2.2.1. 16-bit sigma-delta analog-to-digital converter

The SD16_A is a multichannel 16-bit ΣΔADC implemented in the MSP430FG479. The module consists of one ΣΔADC and an internal voltage reference generator (1.2V). The ΣΔADC has eight fully differential multiplexed analogue input pairs, of which 5 analog input pairs (A0-A4) are available externally on the device. The inputs to channel 5 link to a divided supply voltage, which is used to monitor battery voltage. The inputs to channel 7 are short-circuited internally, and are used to calibrate the offset of the SD16_A input stage.

The conversion mechanism of ΣΔADC is very different from successive-approximation-register (SAR) type ADCs, which was used for analog-to-digital conversion in previous designs of electrochemical instrument[11, 15]. The sigma-delta converter is ideal for low-frequency precision current sensing application for several reasons. First, sigma-delta converters have the innate advantage of attaining high resolution (16-24bits) without special calibration. Second, SD16_A has fully differential inputs, which allow voltage difference between the positive input and the negative input being converted. With differential input range being ±0.5V, SD16_A enables measurement of positive and negative current. Third, ΣΔADC oversamples the desired signal at a much higher rate than the effective bandwidth, thus an expensive anti-aliasing filter with steep roll-offs is not required at the analog input ends. The oversampling nature of the ΣΔADC also average out random noises at the inputs, which increases the signal to noise ratio. However, ΣΔADC trade speed for resolution, with 32KHz clock speed and 512 over-sampling-ratio (OSR), the effective sample rate is only 62.5Hz. Since, the current signal in amperometric detection is at reasonably low frequency (i.e. data will be recorded at 1Hz), it is possible to take full advantage of the 16-bit sigma-delta converter module implemented in MSP430FG479 device.

The built-in 1.2V reference can be used externally as well as internally. When used off-chip, it can supply currents up to 1mA. In the potentiostat circuitry, this regulated voltage is utilized to implement negative potential control and negative current sensing.

2.2.2.2. Low Power Operating Modes

The MSP430 family is designed for ultra-low power applications, and dramatic power saving can be achieved with its 5 software selectable LPMs (reference). LPM3 is probably the most important mode, as it allows the CPU to sleep with a supply current as low as 1.2μA, while the low-frequency auxiliary clock (ACLK) remains active for interrupts and real time clock (RTC) function. In the envisioned application, the MCU spends most of its time in LPM3. It will wake up only when there is an interrupt event, and will restore back to LPM3 upon return from the interrupt service routine.

2.2.2.3. Universal serial communication interfaces

Two USCI modules (USCI_A0 and USCI_B0) are included in MSP430FG479 microcontroller. The USCI_A0 supports the UART mode. It is used to implement USB communication interface with a FT232RL USB to the Serial IC chip (FTDI Ltd., Glasgow, UK). Operated in the synchronous peripheral interface (SPI) mode, USCI_B0 transmits system status, instructions and data to the LCD. The user can navigate the menu displayed on the LCD with four push buttons. When used as a standalone device, the LCD together with push buttons and LEDs constitute the user interface of the system.

2.2.3. Power supply module

As today's world becomes more and more portable, a good power module design is a key part of every portable system design success. Due to its stable performance and light weight, a standard 3V lithium coin cell battery (220mAh) is used here as a low-noise energy source for precision amperometric detection. Extending battery life begins with strategic selection of each system component, followed by careful hardware power management. Additional issues emerge when designing the power section of mixed signal system. Here, several design trade-offs were made to decrease the system power consumption in both active and sleep mode, while keeping the analog power supply noise under control.

2.2.3.1. Self-powered USB communication interface

In active mode, power consumption in the proposed system is dominated by the USB interface module, in particular, the FT232RL chip, which requires 15mA supply current when active (Figure 2.2). Dramatic power saving can be achieved by powering the FT232RL chip through a PC's USB port such as done for many electronic devices used in daily life including MP3 players and xxx. As the USB interface is only active when it is connected to a PC, it is straight forward to power it by the USB port directly, while having all other components running off of the battery.

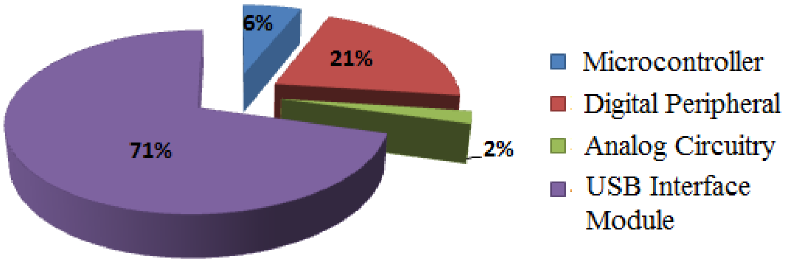


Figure 2.2: Percentage power consumption when the system is in active state.

2.2.3.2. Voltage regulator vs. direct battery connection

Voltage regulators are often used in battery-powered electronic systems to convert constantly decreasing battery voltage to a relatively stable level[13]. However, those step-up regulated charge pumps generate output noises and ripples, usually with peak-to-peak amplitude of tens of milliamperes[16]. For sensitive analog circuitry, noises in power supply rails can easily couple into its signal path, which would cause significant noise in the current reading of the actual redox reaction. Therefore, it was decided to connect the battery directly to the microcontroller and other ICs, without any kind of voltage regulator in between. This will considerably reducing the source noise and allow the measurement of picoampere current response.

At the same time, according to the typical discharge characteristics of a 3V lithium coin cell battery, more than eighty percent of the stored chemical energy can be utilized without having output voltage dropping below 2.8V[17]. Since a minimum 2.5V supply voltage is required for the MCU to conduct all its functions properly, the built-in supply voltage supervisor (SVS) in MSP430FG479 is set to signal the low battery warning when voltage drops below 2.8V.

2.2.3.3. Analog power switch

Most embedded applications spend significant periods of time in standby mode. Therefore standby power consumption is a major factor which determines battery life. To maximize power efficiency, further power saving of the system was achieved by adding a high-side power switch MAX891L (Maxim Integrated Products) to turn off voltage rails for the entire analog circuitry when not in use. So before experiment starts, an I/O port is used to turn on MAX891L, which then supplies 3V power to the potentiostat within 200µs. Before entering LPM, power is cut off from all the ICs in the analog part of the system by simply turning off the switch.

Table 2.1: Comparison of system power consumption in active and idle state.

Components' Category	**Component**	**Supply Current (Active)**	**Supply Current (Idle)**
Microcontroller	AVcc+DVcc	51 µA~420µA	1.2 µA
	SD16_A	800 µA	
	SVS	10 µA	
	DAC12	50 µA	
Digital Peripheral	2 LED	<2mA	
	LCD	<2.5mA	1.5 µA
Analog Circuitry	MAX891L High-Side Power Switch	13 µA	0.1 µA
	MAX1044 Voltage Converter	30 µA	
	INA121 Instrumentation Amplifier	450 µA	
	MAX4634 Analog Multiplexer	<1 µA	
	MAX418 Operational Amplifier	4 µA	
Total Power Consumption		**<6.3mA**	**>2.7 µA**

The power consumptions of all the ICs in the system are summarized in Table 2.1. Supply current for each device is obtained from the typical power consumption in the datasheet, therefore is only used for power budget estimation. With 80% of stored energy in a cheap 3V, 220mAh lithium battery, the system allows more than 28 hours of continuous operation, which equals to 1700 60-second amperometric experiments. As the system spends most of its time in LPM with periodic bursts of activity, with ultra-low supply current, the system can standby for years with one single battery, which makes it ideal for in-field measurements.

2.2.4. Potentiostat

Figure 2.3 shows the block diagram of the potentiostat design. All of the circuit components are carefully chosen from low noise, high accuracy, and low power off-the-shelf ICs.

As shown in Figure 2.3 the control signal coming out of the 12-bit DAC goes through a level-shifter circuit first. This level-shifter circuit doubles the control potential swing by shifting the positive only DAC output to cover both positive and negative range. The potential control amplifier receives a control signal from the level shifter and applies a stable potential difference between the working (WE) and reference electrodes (RE).

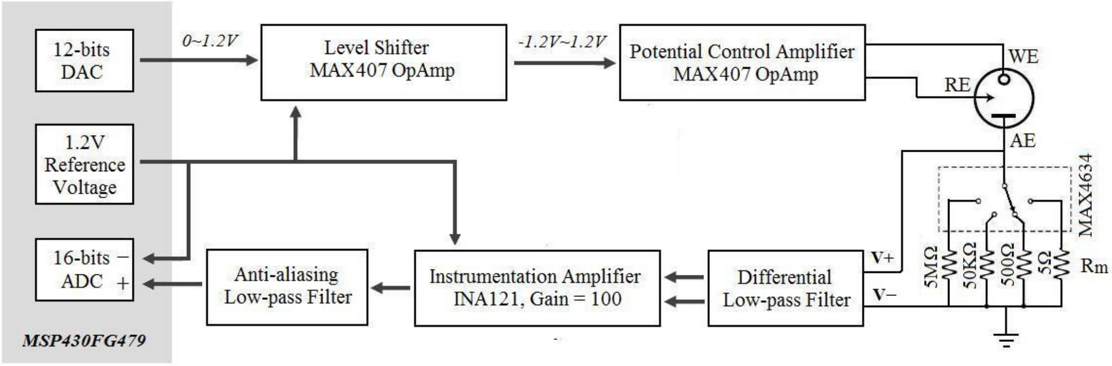

12-bits DAC
1.2V Reference Voltage
16-bits ADC
−
+
MSP430FG479
0~1.2V
Level Shifter MAX407 OpAmp
-1.2V~1.2V
Potential Control Amplifier MAX407 OpAmp
WE
RE
AE
MAX4634
5MΩ
50KΩ
500Ω
5Ω
Rm
V+
V−
Differential Low-pass Filter
Instrumentation Amplifier INA121, Gain = 100
Anti-aliasing Low-pass Filter

Figure 2.3: Block diagram of the 3-electrode potentiostat in the amperometric analyzer.

In this design, current measurement is realized by applying low-side resistive sensing mechanism. A sense resistor is inserted in series with the current path, in our case, between auxiliary electrode (AE) and ground. This resistor produces a small voltage drop, which is then amplified by 100 times using the instrumentation amplifier INA121 (Burr-Brown Corporation, Tucson, AZ). The output voltage feeds into MSP430's 16-bit ADC after proper filtering. As a result, the initial current signal is converted to a differential voltage signal with a V/i gain of 100Rm. Auto-scaling capability is implemented in this design by using MAX4634, a 4-channel low-voltage analog multiplexer (Maxim Integrated Products, Sunnyvale, CA).

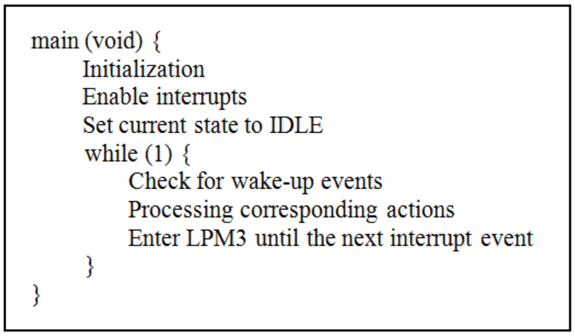
With this circuit topology, the following specification can be achieved: the redox potential application ranges from -1.2V to +1.2V with 1mV resolution and current measurements from ±(1pA to 1mA) in four ranges.

2.3 Software design of the amperometric analyzer

All software is written in C. The development work of the system is carried out with Code Composer Studio (CCS) v4 platform, which is an integrated development environment (IDE) based on the Eclipse open source software framework, and is specifically designed for various Texas Instrument’s devices, including the MSP430 microcontroller family[18].

Constant potential amperometry is the simplest amperometric detection method, which is widely used in glucose monitoring systems as mentioned above. In this study, the amperometric analyzer is programmed to perform amperometry by default. To support and simplify future application upgrades and system integration, a modular and flexible implementation of the embedded software is used.

2.3.1. Event-driven real-time system software



```
main (void) {
    Initialization
    Enable interrupts
    Set current state to IDLE
    while (1) {
        Check for wake-up events
        Processing corresponding actions
        Enter LPM3 until the next interrupt event
    }
}
```

Figure 2.4: Pseudo code for the event-driven software of the system.

In embedded systems, software plays a major role in managing system power consumption, as all hardware usages are managed through software. To ensure sufficient battery life while maintaining performance, an event-driven software architecture is implemented for the amperometric analyzer[11].

Figure 2.4 is a pseudo code example for an event-driven program. After the initialization process, the MCU stays in "sleep" mode as long as there are no events that require CPU intervention. The average current consumption of the system is proportional to the area under the charge utilization profile (Figure 2.5), and thus depends on four independent parameters, specifically, duration in the active and sleep modes, and current usage in both modes. An event-driven software design minimizes the time in active mode, thus to help the system to meet the strict power budget.

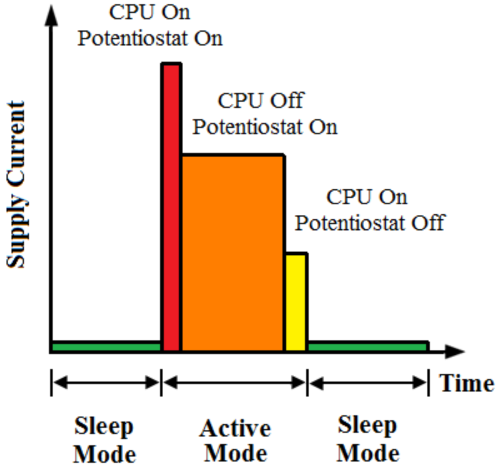


Figure 2.5: Charge utilization profile showing the average current consumption of the system depends on durations in the active and sleep modes, and current usages in both modes.

Table 2.2: System tasks and their corresponding clock signals.

Task	Clock Source	Frequency
CPU	MCLK	1M Hz
Real time clock	ACLK	32K Hz
Interval timer	ACLK	32K Hz
UART for USB communication	ACLK	32K Hz
SPI for LCD control	SMCLK	1M Hz
Analog to digital conversion	ACLK	32K Hz
Flash memory operation	MCLK	1M Hz

To further cut down the power usage of the system, two clock frequencies were used interchangeably based on the task being performed. Tasks that require a fixed number of instruction cycles are executed with the fastest available clock (MCLK and SMCLK at 1M Hz), as a result, CPU on-time is greatly reduces. For those tasks that require a fixed time to complete, the slowest available clock (ACLK at 32K Hz) is used, so required supply current is minimized (Table 2.2).

2.3.2. Finite state machine

Real-time system software implementation can be greatly simplified using the finite state machine (FSM)[11]. Figure 2.6 shows the relationship between 6 states in the current system with their transition events illustrated. The logic behind each transition is very straightforward. The user interface consists of four push buttons and a graphic LCD, is implemented based on this FSM. The system is also configured to operate under PC-control when needed. Test data can be transferred to a PC easily for further analysis. The fool-proof design allows users to fully navigate the menu displayed on the LCD and operate the device with minimum training time.

Specifically, after entering the main loop, the system first enters IDLE state. In this state, all system components are powered down except for the basic timer, which is used to keep the real-time clock. The system enters MENU state when an interrupt is generated by pressing the “ENTER” button. In this state, the MCU wakes up to check the battery and memory status, update the LCD with a user menu displayed. Three control options are available, specifically, test, setup and logs. The user can choose to either start a new experiment, update the system setting or manage old files by pressing different buttons. A low battery or low system memory warning will also be displayed when appropriate.

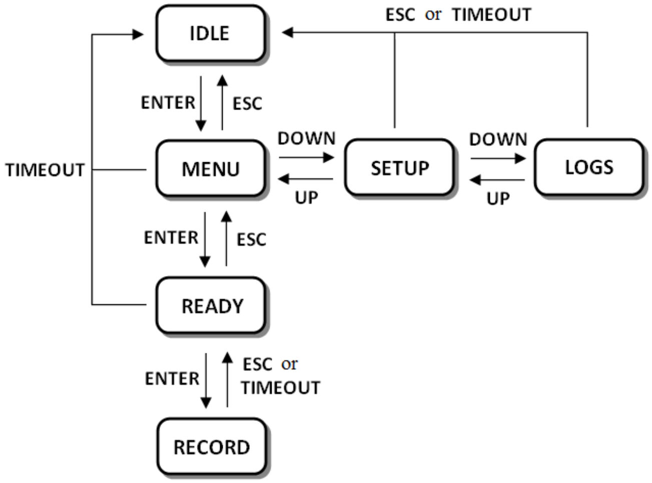


Figure 2.6: The relationship between six states in the finite state machine with their transition events.

In the READY state, the analog circuitry is first powered on, and then a re-initialization sequence of disabled analog peripheral is executed before the system is ready to perform new experiment. This sequence takes about xxx seconds. The RECORD state can be activated with another ENTER button press. Before recording starts, an empty data file is created to store the test result with the test start time as the file name. An amperometric experiment begins by enabling DAC conversion and applying a constant redox potential between WE and RE of the electrochemical sensor. The MCU stays in LPM0 after enabling the DAC, and only wakes up by timer interrupt at each preset interval to perform current recording and data logging tasks. Upon finishing the experiment, the current measurement is displayed on the LCD and the system returns to the READY state. The user can exit the RECORD state anytime during the active experiment and return to the READY state by pressing the ESC button. If the user forgets to turn off the device after use, the system will automatically return to IDLE state after 5 minutes of inactivity. At the same time, all data will remain stored and nothing is lost due to the automatic turn-off.

The basic structure and logic of the embedded software in this work were based on the software design in the miniEC system[11], which was developed in our lab several years ago. Changes have been made in detailed code implementation to adapt the new generation microcontroller, added potentiostat functionality and optimized power usage.

2.4 Result and Discussion

A prototype of the aforementioned amperometric analyzer is shown in Figure 2.7, with an overall dimension of 2.6 inch by 3.2 inch. The ultra-low power design was evaluated using a commercial Multimeter (HP34401A), and the supply current is measured to be 1.35 to 4.3mA in active mode, which corresponds to over 3000 60-second amperometric experiment with a single 3V battery. In sleep mode, a 65μA supply current is measured. Though it is considerably higher than the design estimation of 2.7μA, 113 days of battery life can be achieved in stand-by mode. This increase in energy consumption is probably due to residue current consumption of the already switched-off analog circuitry plus leakage currents occurring throughout the board.

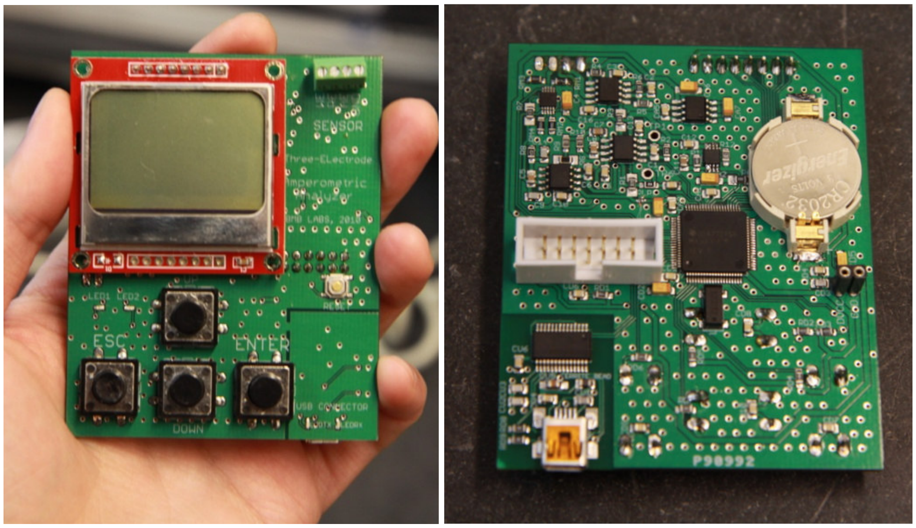


Figure 2.7: (a) Front view and (b) back view of the amperometric analyzer.

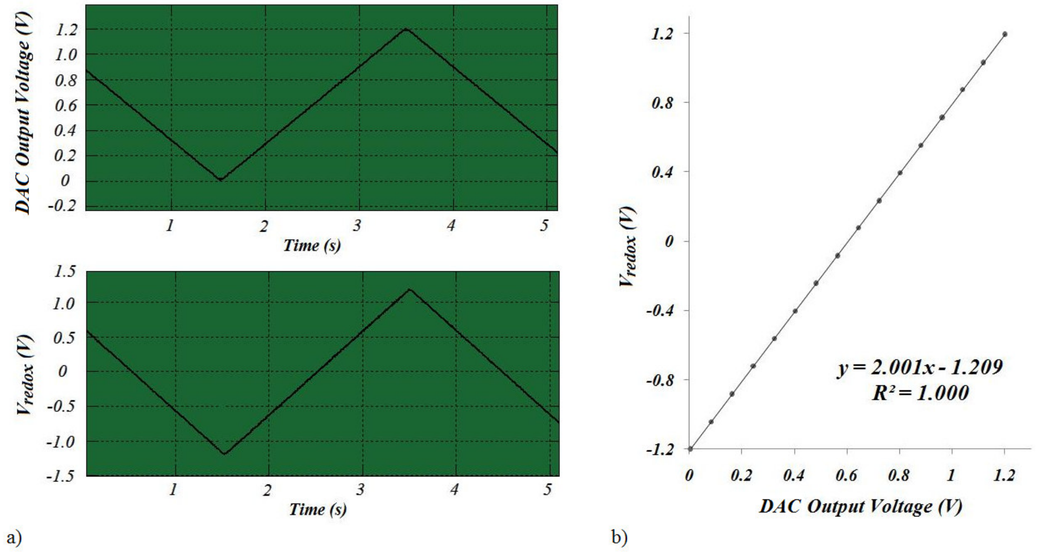


Figure 2.8: (a) Simultaneous measurements of DAC output and the V_{redox}. (b) V_{redox} vs. DAC output voltage.

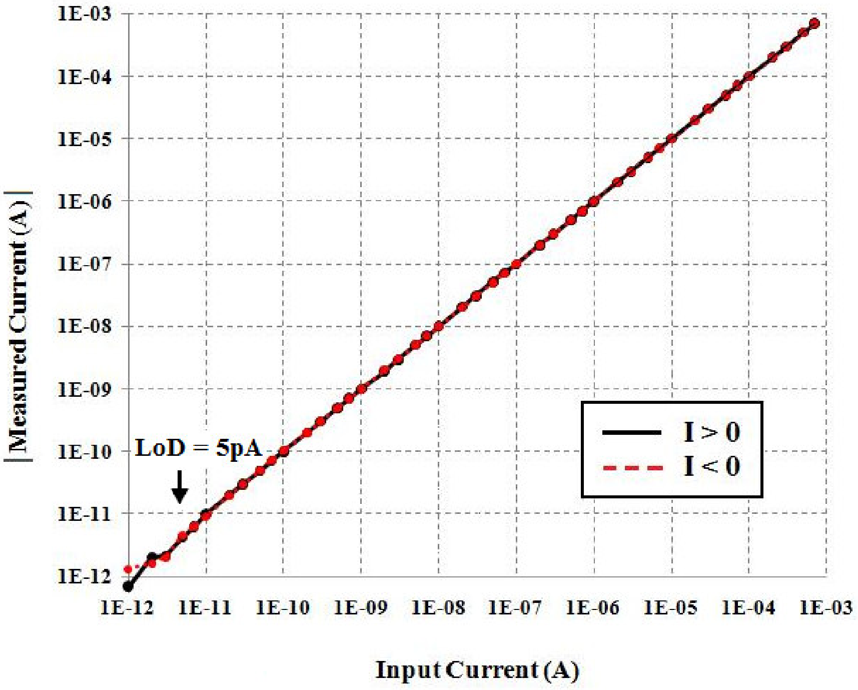


Figure 2.9: Absolute value of current value measured using the amperometric analyzer vs. the input current from the Keithley 4200 device. The LoD of the system is calculated to be +5pA for positive current and -5pA for negative current.

To characterize the potential control function of the system, potential difference between WE and RE (V_{redox}) is measured and is plotted versus the corresponding DAC output (Figure 2.8). Excellent linearity is observed over the whole potential control range with an only -1.2mV system offset, which can be corrected easily in the software if desired.

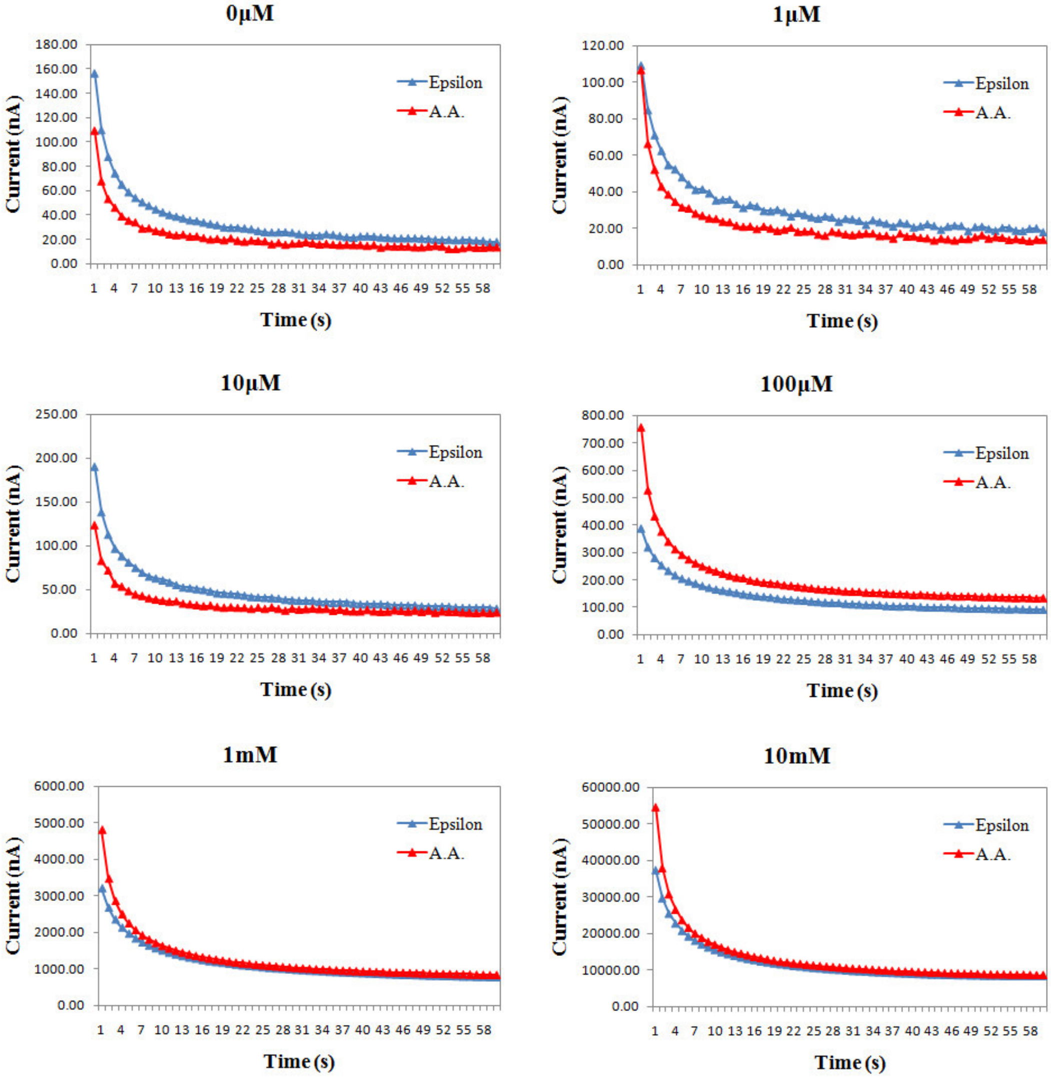


Figure 2.10: Profile of the current signal measured by the amperometric analyzer and Epsilon electrochemical workstation at 1s intervals for a potential of 400 mV applied for a duration of 60 s. Solutions containing varying concentrations of ferro- and ferrihexacyande (0 – 10 mM) in 0.1 M phosphate buffer were detected.

The current measurement performance of our wide range potentiostat is accessed using the Keithley 4200 device characterization system. To provide an accurate input current to the amperometric analyzer, the Keithley 4200 was connected to the AE terminal of the potentiostat through a shielded cable, and its input current was swept from ±(1pA to 1mA) in logarithmic scale. As shown in Figure 2.9, highly accurate current sensing over 8 orders of magnitude is achieved with the amperometric analyzer. After calibration, the experimental results demonstrate that the system has an average 0.17% accuracy in measuring sensor current from ±(10pA to 1mA), and an average 13.56% accuracy when measuring current from ±(1pA to 10pA). If S/N ratio = 5 is used to determine the limit of detection (LoD) of the system, the resultant LoD is calculated to be 5pA for both positive and negative current (Figure 2.9). Selecting the lowest current range, the resolution of the system is measured to be 0.26pA, which is slightly higher than the design expectation (0.1pA).

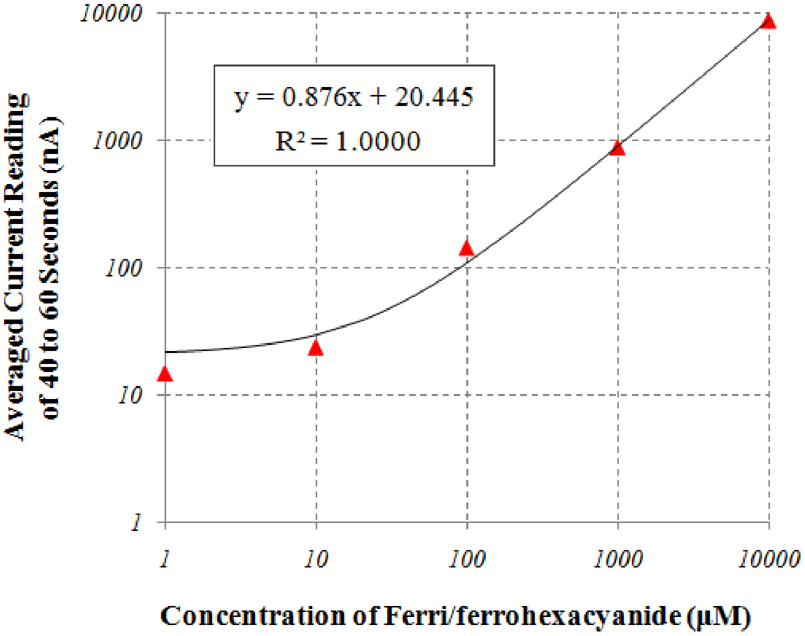


Figure 2.11: Calibration curve of ferri/ferrohexacyanide obtained using the amperometric analyzer. Data are plotted in log-log scale.

The amperometric analyzer was also connected to a standard 3-electrode cell immersed in 0.1M phosphate buffer (pH7.4) with 0μM, 1μM, 10μM, 100μM, 1mM and 10mM ferri/ferrohexacyanide as the sample analyte. In the experiment setup, a planar platinum electrode (3mm diameter), a platinum wire electrode, and an Ag/AgCl electrode were used as the WE, AE and RE respectively. A 400mV redox potential was applied to the 3-electrode cell through the potentiostat, and data were collected at 1 second interval for a duration of 60s.

The experimental result of the system is compared with Epsilon BAS electrochemical workstation using the same standards solutions (Figure 2.10). Excellent agreement was attained over the whole measurement range, which proofs the excellent potential control and current sensing abilities of the amperometric analyzer. A calibration curve of the ferri/ferrohexacyanide is also plotted in log scale using the average current reading of the last 20s of each experiment (Figure 2.11). Due to the 20.445nA interception, the best-fitted linear regression line appears curved below 100nA. A perfect R-square value indicates highly accurate concentration measurement using this system, especially when the wide concentration span of ferri/ferrohexacyanide in this experiment is considered.

2.5 Conclusion

The main objective of this study was to achieve a low power, high accuracy, and wide adaptable amperometric detection with a palm-sized embedded amperometric analyzer. Such a miniaturized embedded system has been designed and developed using off-the-shelf components which together cost less than $70 in prototype quantities. This particular design has very low power consumption and is optimized for field-portable applications. With a single 3V lithium coin cell battery, the system can perform over 3000 60-second long amperometric experiments or standby for about 4 months. Using the new MSP430FG479 microcontroller, which is equipped with a 16-bit ΣΔADC, accurate current detection spanning 8.5 orders of magnitude (5pA to 1mA) is achieved with this system, with the best measurement resolution being 0.26pA. According to system characterization result, the proposed amperometric analyzer can achieve comparable accuracy as the BAS Epsilon with an ultra-low LoD of 5pA. Only constant potential amperometry is implemented with the current version of the embedded software, however, the instrument can be easily upgraded with new electrochemical techniques by adapting only the embedded software.

REFERENCES

1. Sawyer, D.T., *Experimental Electrochemistry for Chemists*. 1974: John Wiley & Sons Inc.
2. Bard, A.J. and L.R. Faulkner, *Electrochemical methods: fundamentals and applications*. 2nd ed. 2001, New York: John Wiley & Sons, Inc.
3. Avdikos, E.M., M.I. Prodromidis, and C.E. Efstathiou, *Construction and analytical applications of a palm-sized microcontroller-based amperometric analyzer.* Sensors and Actuators B: Chemical, 2005. **107**(1): p. 372-378.
4. Andreescu, A. and A. Sadik, *Trends anc challenges in biochemical sensor for clinical and enviromental monitoring.* Pure and applied chemistry, 2004. **74**(4): p. 861-878.
5. Prodromidis, M.I. and M.I. Karayannis, *Enzyme Based Amperometric Biosensors for Food Analysis.* Electroanalysis, 2002. **14**(4): p. 241-261.
6. Ping, J., et al., *A Prussian blue-based amperometric sensor for the determination of hydrogen peroxide residues in milk.* Ionics, 2010. **16**(6): p. 523-527.
7. Stetter, J.R. and J. Li, *Amperometric Gas SensorsA Review.* Chemical Reviews, 2008. **108**(2): p. 352-366.
8. Fort, A., et al., *A two electrode C-NiO Nafion amperometric sensor for NO2 detection.* Microelectron. J., 2009. **40**(9): p. 1308-1312.
9. Trnkova, L., et al., *Amperometric Sensor for Detection of Chloride Ions.* Sensors, 2008. **8**(9): p. 5619-5636.
10. Kwakye, S., V.N. Goral, and A.J. Baeumner, *Electrochemical microfluidic biosensor for nucleic acid detection with integrated minipotentiostat.* Biosensors and Bioelectronics, 2006. **21**(12): p. 2217-2223.

11. Kwakye, S. and A. Baeumner, *An embedded system for portable electrochemical detection.* Sensors and Actuators B: Chemical, 2007. **123**(1): p. 336-343.

12. Gopinath, A.V. and D. Russell, *An inexpensive field-portable programmable potentiostat.* The Chemical Educator, 2006. **11**(1): p. 23-28.

13. *MSP430F47x Mixed Signal Microcontroller Datasheet*. 2008, Texas Instruments.

14. *MSp430x4xx Family User's Guide*. 2010, Texas Instruments.

15. Shi, Y., et al., *Design and fabrication of a miniaturized electrochemical instrument and its preliminary evaluation.* Sensors and Actuators B: Chemical, 2008. **131**(2): p. 516-524.

16. *TPS0204 Datasheet*. 2001, Texas Instruments.

17. *Energizer No. CR2032 Engineering datasheet*, Everyday Battery Co.

18. *Code Composer Studio v4.2 for MSP430 User's Guide*. 2010, Texas Instrument.

Chapter 3 Design and characterization of a wide dynamic range, pico-ampere resolution potentiostat for field-portable electrochemical applications

3.1 Introduction

The ability to measure a specific analyte concentration is of crucial importance in many field monitoring tasks, such as toxic agent screening[1, 2], breath alcohol testing[3], and home-care patient monitoring[4-7]. With its inherent sensitivity and portability[8], amperometric sensors offer great promise for quantitative detection of a wide variety of electrochemically reactive organic, inorganic and biochemical analytes. With continuous introduction of various electrochemical based sensors[1, 9, 10], there is great need for a widely adaptable amperometric analyzer with miniaturized size, decreased cost but comparable performance with conventional “bench-top” electrochemical workstations[8].

In amperometric assays, the reduction-oxidation (redox) potential of the working electrode (WE) is adjusted to a preset value, so that once the redox active species reach the WE surface, they immediately undergo electron transfer reactions[11]. In an unstirred solution, at steady state, the rate of electron transfer is diffusion-controlled and is directly proportional to the concentration of the reactant[12]. Therefore, this steady state current is used as an accurate measure of the analyte concentration in the sample.

A potentiostat is the electronic interface for an amperometric sensor. The three-electrode potentiostat, first built by Hickling in 1942[13], is used to monitor amperometric assays with enhanced control and measurement accuracy. In a two-electrode system, the counter electrode (CE) is used to simultaneously maintain a constant interfacial potential at WE, while completing the circuit to allow charge to flow through the cell[11, 12]. However, control and measurement errors arise due to the fact that the two functions of the CE are sometimes mutually exclusive[11]. In a three-electrode amperometric assay, this problem is overcome by dividing the two functions of the CE between a reference electrode (RE) and an auxiliary electrode (AE) [11]. With the AE supplying a current flow in the opposite direction to balance the redox current at the WE, the RE is used solely as a potentiometric probe (zero current flow) to establish an accurate potential reference point (E_{RE}) against which the WE potential (E_{WE}) is assessed[11].

With comprehensive discussion and comparison, Greef (1978) and Ahmadi (2009) reviewed various poteniostat topologies based on their potential control and current measurement techniques [14, 15]. Two main potential control configurations, specifically grounded-WE and grounded-AE, and two common current measurement methods, i.e. transimpedance amplifier and resistive current sensing, were widely used in previous potentiostat circuit design articles as well as commercially available potentiostat units[14, 15]. As each architecture has its own advantages and disadvantages, successful potentiostat designs are always based on specific application requirements and design constrains.

A self-contained amperometric analyzer based on the new MSP430FG479 microcontroller is developed in our lab to conduct amperometric detection with various electrochemical sensors. Constructed using off-the-shelf IC chips and powered by a single 3V battery, the potentiostat circuitry inside the amperometric analyzer is specially designed to allow accurate redox potential control from -1.2V to +1.2V with 1mV resolution, and current measurement ranges from ±(1pA to 1mA) covered by four selectable detection levels. In this paper, this novel potentiostat design is presented with a special focus on components selection methodology, which allows ultra-sensitive, bidirectional, wide span potential control and current sensing in a portable embedded system.

3.2. Potentiostat circuit topology

Figure 3.1 shows the schematic of the potentiostat design. This circuit has two functional blocks, a potential control block and a current measurement block. Together they form a 3-electrode potentiostat. The design of this electrochemical interface enables wide control potential swing, large current measurement range including pico-ampere scale, great stability unaffected by sensor impedance, and low power consumption compatible with 3V power supply.

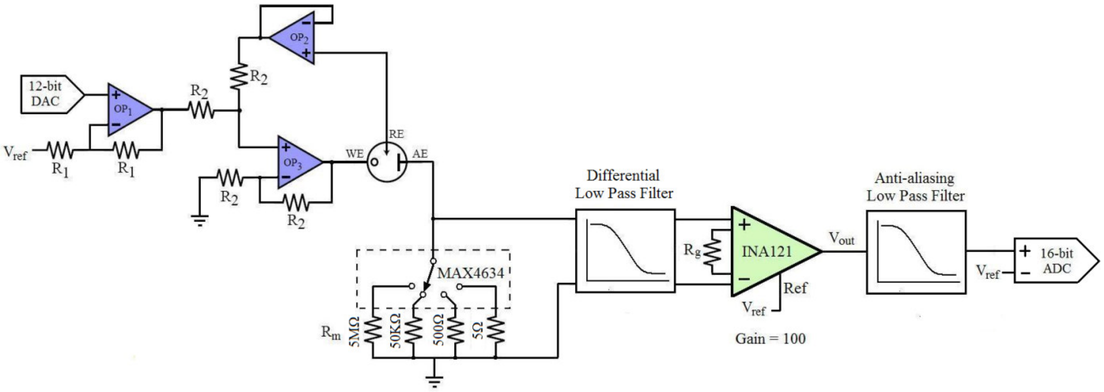


Figure 3.1: Schematic of the 3-electrode potentiostat.

3.2.1. Potential control circuit block

The potential control block of the potentiostat is shown in Figure 3.2(a). This circuit receives a control signal from 12-bit DAC of the MCU and applies a stable potential difference between the working (WE) and reference electrodes (RE). The circuit is built with low power, low noise MAX407 operational amplifiers (op-amp) (Maxin Intgrated Products, Sunnyvale, CA). This op-amp has low input bias current

(<0.1pA), low offset voltage (max 500μV), and rail-to-rail output swing, which make it ideal for potentiostat circuit. Moreover, this micropower precision op-amp draws only 1.2 μA max quiescent current[16], and is thus very suitable for portable electronic applications.

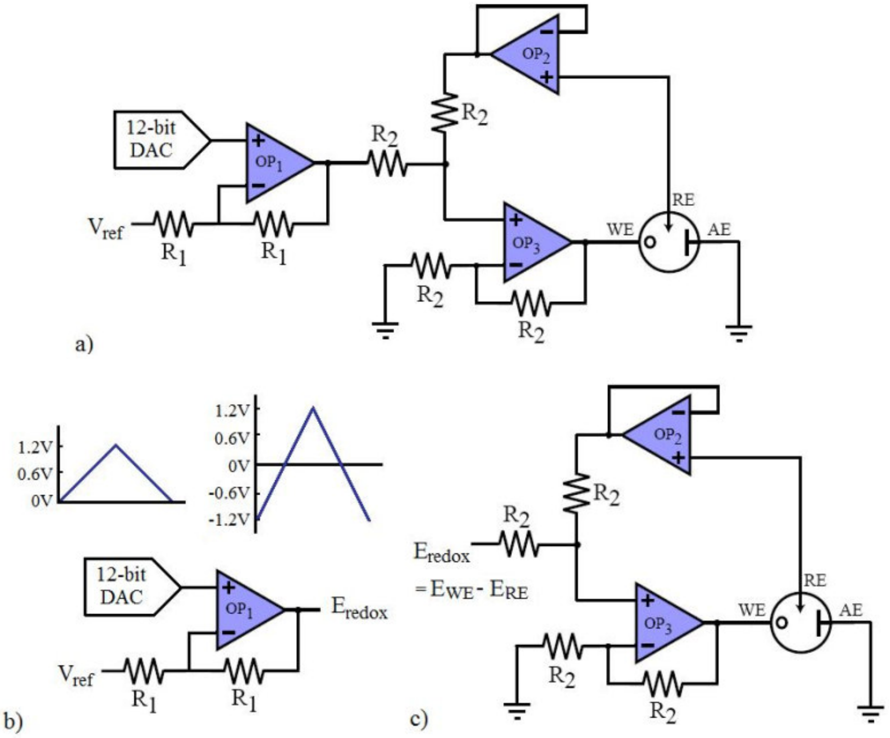


Figure 3.2: Schematic of (a) the overall potential control block of the potentiostat, (b) level shifter circuit, (c) potential control circuit.

The term operational amplifier, abbreviated op-amp (OP), was created in the 1940s to refer to a special kind of amplifier that can be configured to perform a variety of mathematical operations. Op-amps are important building blocks for a wide range of electronic circuits. The ideal model of op amp was based on three simple assumptions: 1) gain is infinite, 2) input resistance is infinite, and 3) output resistance is zero. Applying these assumptions results two fundamental rules, which are used to guide design process of all kind of op-amp circuits. The first rule, the inverting and non-inverting inputs conduct no current to op-amp, i.e. $\mathrm{i_{in}} \approx 0$. The second rule is, when in negative feedback configuration, the op-amp output voltage equals to the voltage at its input ends, i.e. $\mathrm{V_0} = \mathrm{V_+} = \mathrm{V_-}$.

As shown in Figure 2(b), the control signal coming out of the 12-bit DAC goes through a level-shifter circuit first. This level-shifter circuit doubles the redox potential (E_{redox}) swing by shifting the positive only DAC output to cover both positive and negative potential ranges. The conversion is based on following equation

$$\mathrm{E_{redox}} = \mathrm{E_{WE}} - \mathrm{E_{RE}} = 2\mathrm{V_{in}} - \mathrm{V_{ref}}.$$

With V_{in} ranging from 0 to 1.2 V and V_{ref} equaling to 1.2V, the resulting redox potential ranges from -1.2V to +1.2V, which is adequate to meet redox potential requirements for most voltammetric detection applications.

Voltage coming out of the level-shifter circuit is then injected into circuits shown in Figure 2(c). OP2 and OP3 set the WE potential to the value needed, so that the potential difference between WE and RE is maintained to E_{redox}. Applying the two fundamental rules of op-amp operations, it is very straightforward to prove that $\mathrm{E_{WE}} - \mathrm{E_{RE}}$, the potential difference between WE and RE, is equal to E_{redox}.

Stability is an important characteristic of a potentiostat circuit. In many previous potentiostat topologies, especially those with ground-WE configuration, negative feedback control is used to maintain a desired potential between WE and RE[14]. However, being an integral part of the control loop, the electrochemical cell can cause oscillation of the system when the capacitive element in the sensor impedance exceeds certain limits[17]. In our design, having the electrochemical sensor excluded from negative feedback loop, the stability of the system is no longer influenced by the sensor impedance. This is an important characteristic of the design, as we target to make a portable potentiostat that can be used to control and measure a wide variety of electrochemical cells.

3.2.2. Current measurement circuit block

In this design, current measurement is realized by applying a low-side resistive sensing mechanism (Figure 3.3)[18], which is an accurate yet cost-effective method for low current applications (picoamps to milliamps). A sense resistor is inserted in series with the current path, in our case, between auxiliary electrode (AE) and ground. This resistor produces a small voltage drop, which is then amplified 100 times using the instrumentation amplifier INA121 (Burr-Brown Corporation, Tucson, AZ). The output signal, V_{out}, feeds into MCU's 16-bit Σ△ADC after proper filtering. As a result of this circuit, the initial current signal, i_{cell}, is converted to a voltage level, V_{out}, with a V/I gain of 100Rm.

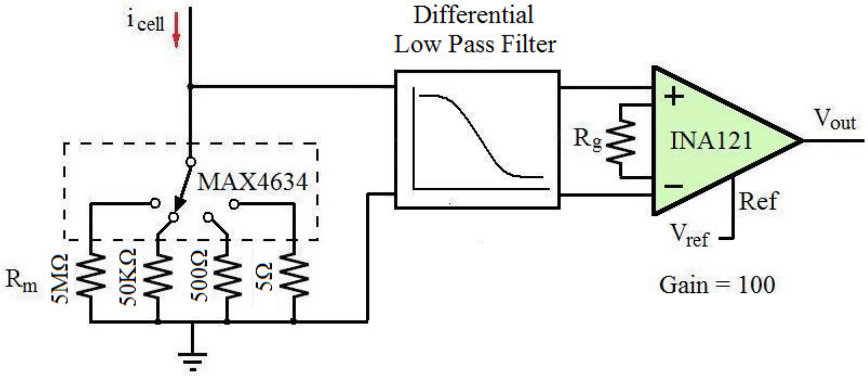


Figure 3.3: Schematic of current sensing block of the potentiostat.

To achieve highest precision over a wide dynamic range of i_{cell}, the circuit is designed to operate over different measurement ranges by adjusting the resistance of the sense resistor Rm. Auto-scaling capability is implemented in this design by using MAX4634, a 4-channel low-voltage analog multiplexer (Maxim Integrated Products, Sunnyvale, CA), i.e. the software automatically selects the maximum Rm through the analog multiplexer without overloading the output of the INA121 instrumentation amplifier.

Choosing the appropriate instrumentation amplifier for signal amplification in a design is critical. For example, with design constrains of our design, such as ±3V power supply, picoampere range current sensing, low noise and low power consumption, only very few off-the-shelf IC chips can meet all these requirements. INA121 was finally picked for its combination of wide supply range (±2.25V to ±18V), low bias current (±4pA), and low quiescent current (±450μA)[19].

The decision of using INA121 leads to two other important design decisions of the potentiostat.

As a LoD of 1pA is targeted, the appropriate resistance value for the largest sense resistor needs to be selected. With its gain set to 100, the input-referred voltage noise of INA121 at 0.1 to 10Hz has a peak-to-peak amplitude of around 0.5μV[19]. As a result, it is impossible to detect low frequency differential signals with amplitudes lower than 0.5μV at the input of INA121. Moreover, for picoampere current sensing, thermal noise of resistors cannot be ignored, as a resistor in the MΩ range has an RMS voltage noise close to μV at low frequency bands[20]. Allowing for some extra room for other noise sources, a 5MΩ sense resistor is picked to generate a 5μV differential voltage with 1pA current.

The second design decision is imposed by the limited common mode input range of INA121. When using ±3V power supply, INA121 can only accept common mode voltage from -0.9V to +1.8V[19]. Although the redox current of the electrochemical sensor can be measured from either WE or AE side in an electrochemical point of view[12], resistive current sensing in this design can only be conducted at the AE side. Because the common mode input voltage at the AE side is always around the ground potential, it meets the requirement of the common mode input range of the INA121.

Table 3.1: Four current sensing levels and their corresponding measurement parameters.

Level	Rm (Ω)	V/i Gain (V/A)	Theoretical Current Measurement Range	Target Current Measurement Range	Linear range	Resolution
A	5	500V/A	±(100nA~1mA)	±(1uA~1mA)	1000	100nA
B	500	50KV/A	±(1nA~10uA)	±(10nA ~10uA)	1000	1nA
C	50K	5MV/A	±(10pA~100nA)	±(100pA~100nA)	1000	10pA
D	5M	500M/A	±(0.1pA~1nA)	±(1pA~1nA)	1000	0.1pA

In this design, 1pA to 1mA redox current is covered by four overlapping current detection levels, each with a dynamic range of 1000 (Table 1). The value of each sense resistor and their corresponding V/i gain are shown in Table 3.1. The theoretical current detection ranges are calculated based on a presumed 13.5 useful bits from the 16-bit Σ△ADC, and their corresponding measurement resolution are set by the corresponding lowest detection limit. In order to achieve the 13.5 useful bits using a 16-bit Σ△ADC, noise level in the signal should be controlled under 2.5-bit wide, which corresponds to a 105μV p-p noise level at the input terminal of the Σ△ADC.

A 1.2V internal generated reference voltage is used for the analog to digital conversion in our design[21, 22]. To ensure conversion accuracy, only the higher portion of the input voltage range of the Σ△ADC is used to implement current measurement, as shown in the target current measurement range column of the Table 1. Moreover, with buffer zones reserved between neighboring levels, the design provides extra-flexibility for the user to choose a detection range based on the resolution requirement of their specific application, as well as to allow easy design of the auto-ranging algorithm[23].

3.3. Signal path filtering strategies

The noise properties are often of paramount importance in analog circuit design. Especially for high precision applications, like picoampere range current sensing in our case, noises may be coupled into signal paths from its input, output, even power-supply pin[20]. Luckily, these error signals are often at high frequency bands, thus can be attenuate by appropriate filtering techniques[20].

3.3.1. Pre-amplifier-filtering stage

As instrumentation amplifiers often have relatively low bandwidth, out-of-band high frequency content can seriously affect the DC performance of the amplifier through radio frequency interference (RFI)[24, 25]. Once signal distortion occurred through RFI, low-pass-filtering at the instrumentation amplifier's output can no longer remove the error[24, 25]. So it is most effective to suppress high frequency pick-up noise before the first stage of signal amplification. Therefore, a differential low-pass filter (LPF), as shown in Figure 3.4, is added to the design before the input ends of INA121[25].

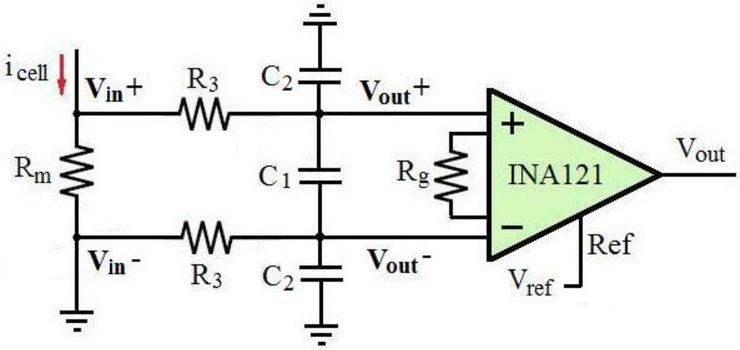


Figure 3.4: Schematic of the differentia low-pass filter for INA121 instrumentation amplifier.

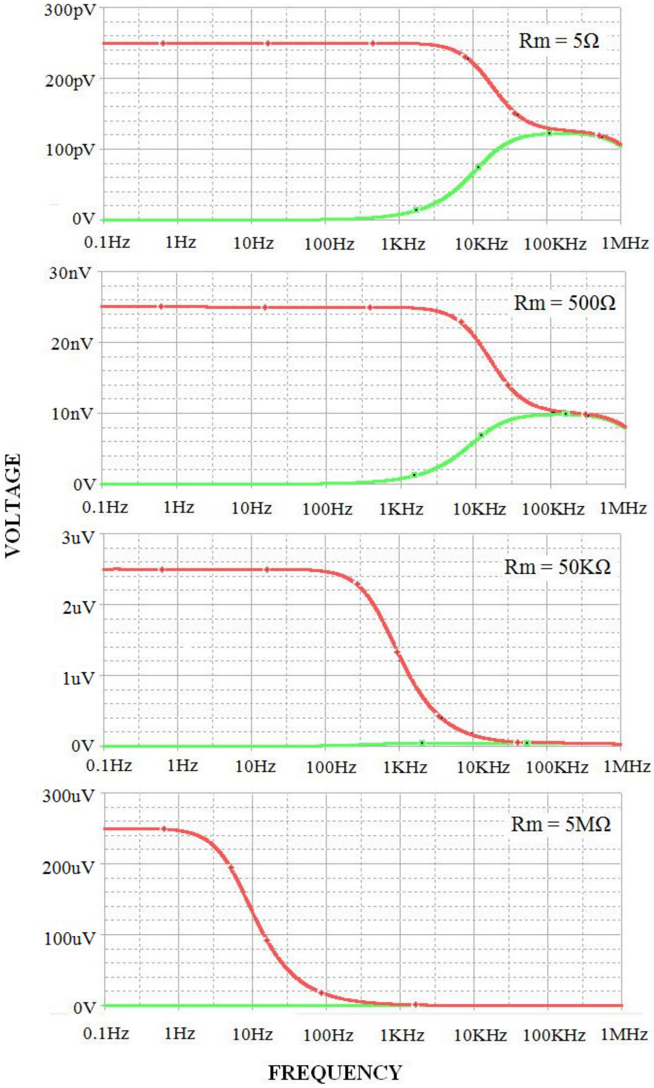


Figure 3.5: PSpice AC simulation result of V_{out}+ (red) and V_{out}- (green) nodes of the differential low pass filter.

Though it is a simple circuit, the design problem becomes complicated because of the interaction between the LPF and the sense resistor Rm. PSpice 9.1 is used to simulate the change in filter response occurring between different sense resistors. Figure 3.5 shows the AC simulation result of the filter response with 50pA AC current injected into the sense resistor. With the resistance of R_m increasing from 5Ω to 5MΩ, the cutoff frequency of the differential LPF decreases dramatically. With R3 = 1K, C1 = 5nF and C2 = 100pF, all cutoff frequencies fall between the sample rate (1Hz) and the 3dB bandwidth of INA121 at gain = 100 (30KHz)[25]. Thus, out-of-band noise can be mostly eliminated with one single differential LPF for all four current sensing levels without reducing their measurement bandwidths.

3.3.2. Anti-aliasing low pass filter

An anti-aliasing low pass filter is included in the design before the analog-to-digital conversion stage (Figure 3.6). The anti-aliasing filter is implemented with a second-order Sallen-Key topology, which gives a steeper roll-off in the stop band than simple RC filter[20].

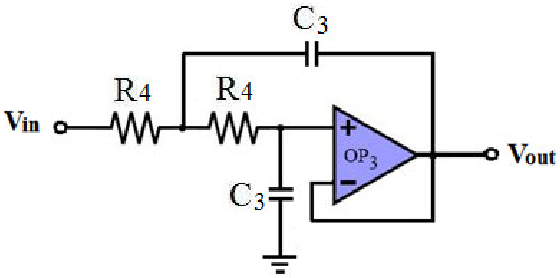


Figure 3.6: Schematic of the anti-aliasing LPF.

The anti-aliasing LPF has two main functions. The first function, as suggested by its name, is to attenuate the higher frequencies (greater than the Nyquist frequency fs/2), thus preventing the aliasing components from being sampled by ADC. With a 32KHz clock and an oversampling ratio of 256, the Nyquist frequency of the Σ△ADC is 62.5Hz. So the cutoff frequency of the LPF needs to be at least 62.5Hz to avoid aliasing[21]. The second function of the filter is to increase signal-to-noise ratios by removing noise peaks that are superimposed on the analog signal, as well as attenuating the quantization noise from Σ△ADC (Figure 3.7)[20, 21, 26]. Therefore, further decreasing the cutoff frequency of the filter is desirable in terms of increasing signal-to-noise ratios.

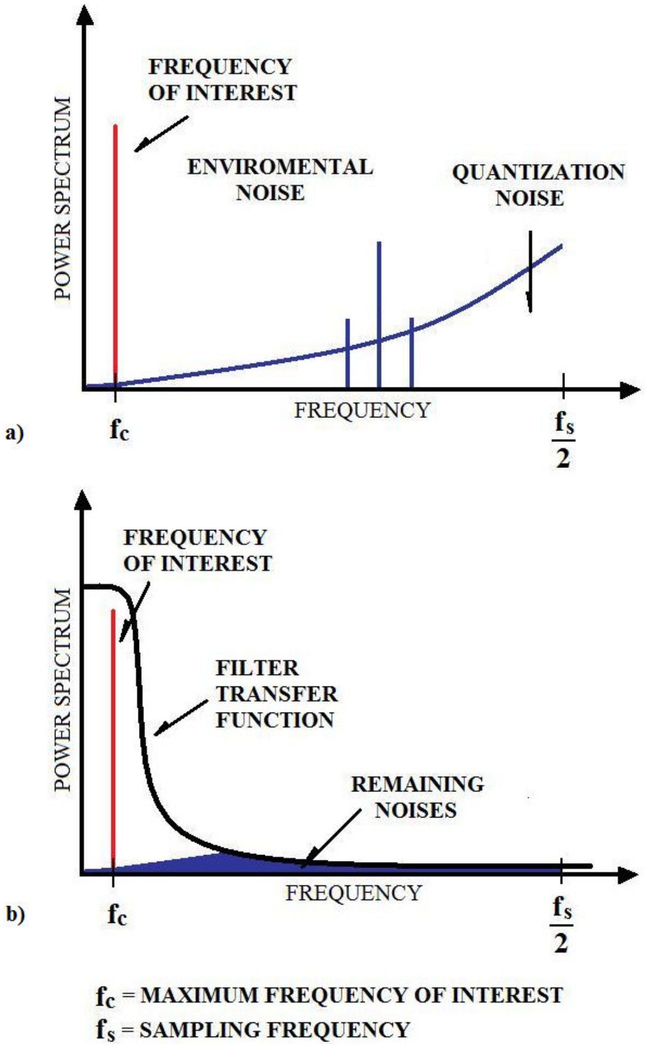


Figure 3.7: (a) Power spectrum of signal together with system noise. (b) System noise is greatly reduced after passing through the anti-aliasing filter.

In the current version of the amperometric analyzer, electrochemical detection is implemented with a maximum sample rate of 1Hz. Therefore, the cutoff frequency of the low pass filter is selected to be 16Hz for enhanced signal-to-noise ratios, however, it can be easily expanded in future versions of the system if desired.

3.4 Result and Discussion

The amperometric analyzer was prototyped on a custom 2-layered PCB. The potential control ability of the potentiostat over the whole control range is accessed by measuring DAC output voltage and the corresponding V_{redox} simultaneously. By plotting the DAC output voltage versus the resultant V_{redox} (V_{WE}-V_{RE}) (Figure 3.8), linear regression analysis demonstrates the high-accuracy bidirectional potential control capability of the potentiostat. When measuring the V_{redox} value using the Tektronix 2245A oscilloscope, voltage noise with peak-to-peak amplitude much less than 1mV is observed at all time, which demonstrate that a 1mV resolution is successfully achieved by the control circuitry.

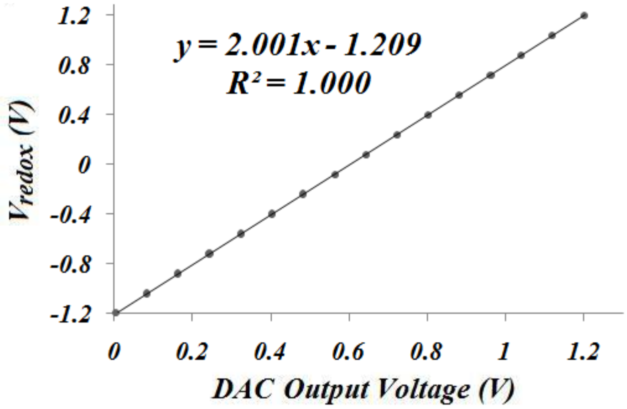


Figure 3.8: V_{redox} vs. DAC output voltage.

To characterize the current sensing capability of the potentiostat circuit, a Keithley Instruments Model 4200 source/meter was used to provide a constant current source for the potentiostat to measure (Figure 3.9). For each current measurement level, the input current was swept in logarithmically fashion to cover a current range spanning 4 orders of magnitude. Input currents that are smaller than the lower design limit (denoted by arrows in Figure 3.9) were also tested, to obtain the true detection limit for each level. The raw ADC output code is normalized by subtracting the system offset and then plotted versus the input current in Figure 3.9. Sixteen parallel readings were recorded for each input current, and the average value is calculated to obtain each data point. Except for a small non-linear region below 5pA, excellent agreement is obtained between design expectations and testing results for both positive and negative current at all gain selections. Thus, through gain selection (level A-D) the potentiostat can conduct accurate current measurement over a very wide range with picoampere sensitivity.

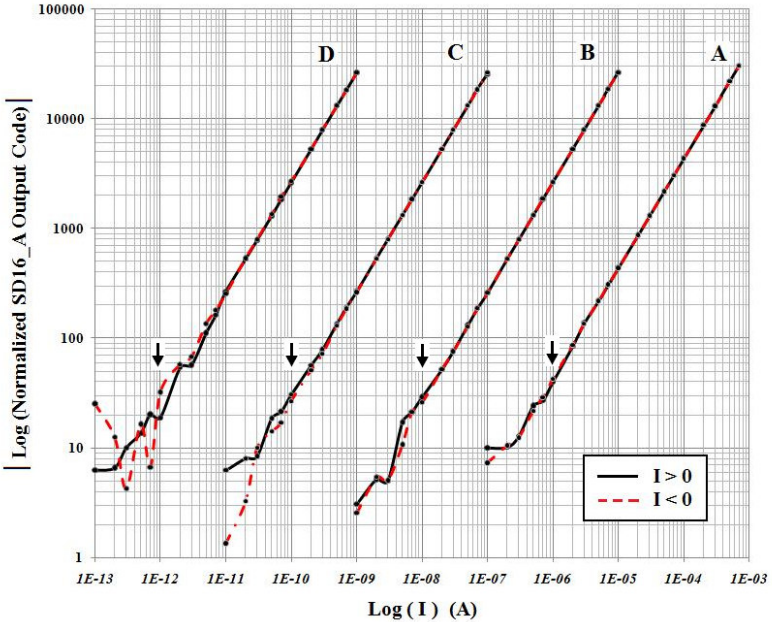


Figure 3.9: The characteristics of the potentiostat at different gains. Gain selection (A-D, Table 3.1) allows the potentiostat to sense redox current over very wide range. Black arrows indicate design estimations of the lower detection limit of each level.

As an example, the testing results of the finest current range (level D) is listed in Table 3.2. Twenty one data points in this current range were measured and analyzed. For current values below the design estimation of the lower limit (1pA), low S/N ratios are observed, as noise dominates these measurement results. For input signal ≥ 10pA, significant enhancement in measurement precision and accuracy is observed. With a S/N=5 cutoff, the detection limit of the system is estimated to be around 5pA, which is slightly higher than the 1pA design target.

Table 3.2: Characterization of current sensing performance of the finest current measurement level (level D). The detection limit of the system is 5pA.

Input Current (pA)	**Measured Current (pA)**	**StDev (pA)**	**S/N**	**% Deviation from input current**
0	0.36	0.20	0.00	n/a
0.1	0.21	0.84	-0.18	110.30
0.2	0.22	0.22	-0.62	11.78
0.3	0.35	0.14	-0.04	18.10
0.5	0.49	1.11	0.12	2.24
0.7	0.74	0.19	1.98	5.31
1	0.82	0.17	2.75	18.15
2	2.02	0.58	2.86	1.22
3	2.40	0.45	4.53	19.96
5	**4.20**	**0.16**	**24.1**	**16.03**
7	6.13	0.21	27.2	12.44
10	10.1	0.15	65.3	0.96
20	20.0	0.14	138	0.01
30	29.7	0.16	185	0.87
50	49.7	0.18	271	0.60
70	69.9	0.08	902	0.20
100	99.7	0.09	1,153	0.26
200	199.7	0.11	1,802	0.14
300	300.2	0.26	1,157	0.06
500	500.0	0.18	2,776	0.01
700	700.1	0.21	3,314	0.02
1000	1,000.0	0.11	9,409	0.00

Similar statistics as shown in Table 3.2 were also performed for current level A to C. Current sensing performances for all 4 current sensing levels are summarized in Table 3.3. In the table, current measurement resolution of each level is calculated based on the maximum standard deviation of all sampled data points. However, exception is made for level A. Because for level A, the measurement standard deviation (19nA max) is so small that it exceeds the 16-bit limit of the ADC, which equals to 31nA. As a result, the 16-bit resolution limit of the ADC is used as the resolution for this current range. As demonstrated in Table 3.3, we successfully achieve 0.26pA sensitivity at level D and a dynamic range of 8.5 orders of magnitude from 5pA to 1mA over 4 detection levels.

Table 3.3: Summary of current sensing performance of each level.

Level	**Useful Current Range**	**Max StDev**	**Max % Deviation**	**Max Deviation as a % of Current Range**	**Resolution**	**Resolution as a % of current range**
A	±(1µA~1mA)	19nA	0.23	±0.0025%	31nA	0.0031%
B	±(10nA~10µA)	0.39nA	1.92	±0.012%	0.39nA	0.0039%
C	±(100pA~100nA)	9.28pA	0.81	±0.009%	9.28pA	0.009%
D	±(5pA~1nA)	0.26pA	16.03	±0.048%	0.26pA	0.026%

Even with such a stringent evaluation criteria, when comparing our potentiostat to two commercially available mini-potentiostat systems, specifically PamSens (Palm Instruments BV, Netherlands) and WaveNano (Pine Research Instrumentation Inc., Raleigh, NC) (Table 1.1), improved measurement resolution and accuracy is achieved in conjunction with a much lower LoD (5 pA vs. 1nA and 100pA).

Amperometric detection of ferri/ferrohexacyanide in phosphate buffer (pH7.4) is performed using our potentiostat as well as the BAS Epsilon. A standard 3-electrode electrochemical cell was constructed using a planar platinum electrode (WE), a platinum wire electrode (AE) and a Ag/AgCl electrode (RE) (Figure 3.10(a)). The three electrodes were connected to our potentiostat through a shielded 3-connector wire, while it is connected to the Epsilon using a coaxial cable. In order to access current detection ability of the potentiostat across different current levels, 0µM, 1µM, 10µM, 100µM, 1mM and 10mM ferri/ferrohexacyanide standard solutions were prepared after a series 10-fold serial dilutions. For each amperometric test, 400mV redox potential was applied for 60 second and resultant redox current was sampled at 1Hz. Three parallel experiments were performed at each analyte concentration using our potentiostat as well as the Epsilon.

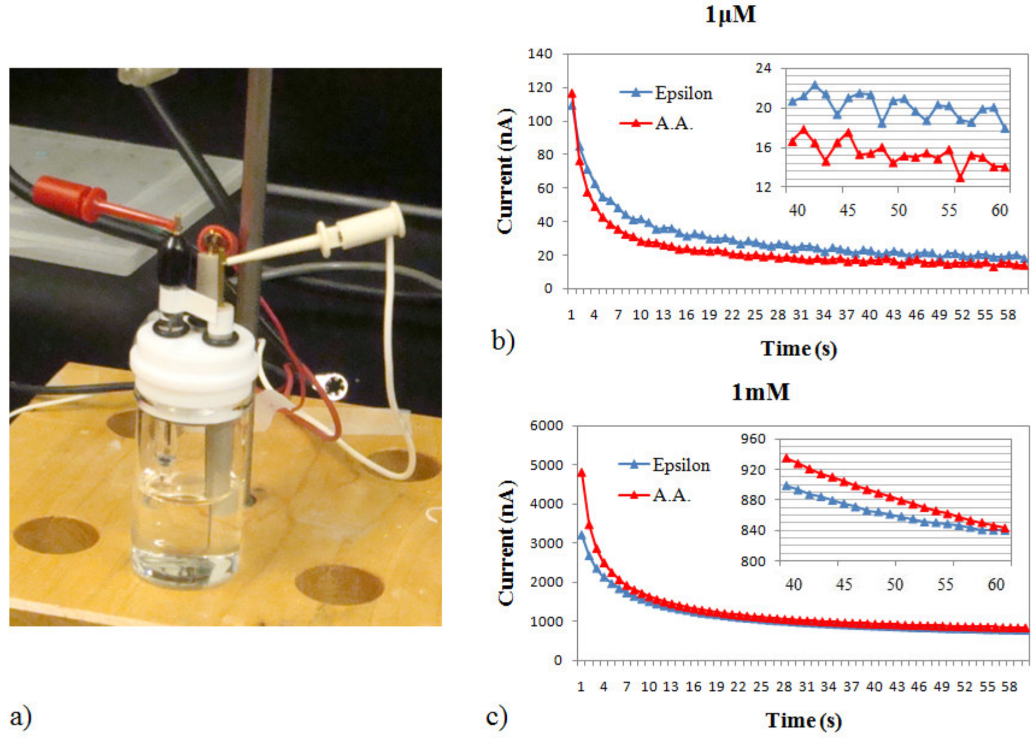


Figure 3.10: (a) Experimental setup of the 3-electrode amperometric detection. (b) and (c) are current profile of 1μM and 1mM ferri/ferrohexacyanide respectively, with enlarged views showing current readings of 40 to 60s.

The only significant noise perturbation event occurred during the measuring of 1μM ferri/ferrohexacyanide solution (Figure 3.10(b)). Noise with a peak-to-peak amplitude of 5nA and an oscillation period of 5 seconds was observed in current profiles obtained using both systems (Figure 3.10(b)). Since the measured noise exist in detection results of both systems, it is probably picked up by the electrochemical cell instead of the potentiostat circuit. Except for the 1μM case, smooth decay of redox current with time, which is the signature current profile for constant-potential amperometry, is observed in all remaining testing results (Figure 3.10 (c)).

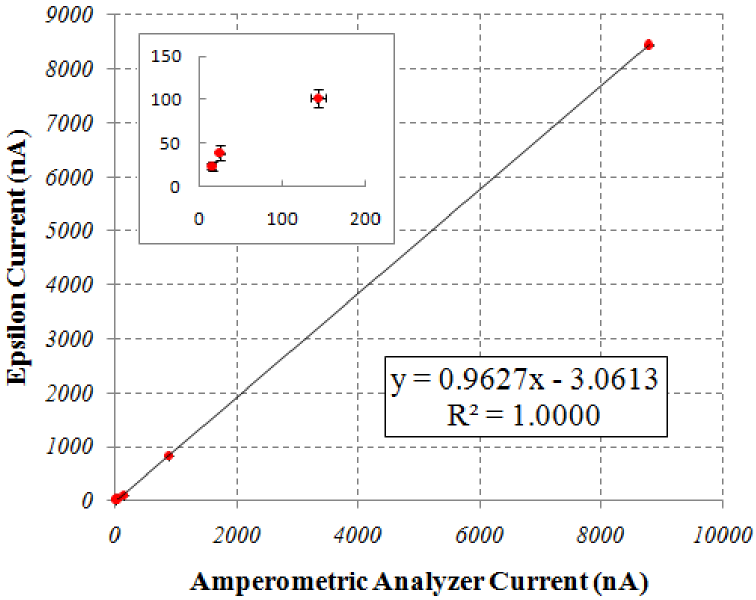


Figure 3.11: Correlation plot comparing the averaged current measurement obtained using Epsilon and the amperometric analyzer

Highly consistent redox current measurements were observed between the two systems and across triplicate analyses. Figure 3.11 shows the correlation plot comparing the averaged current measurement obtained using the BAS Epsilon and our potentiostat. A linear regression analysis of the association resulted in a slope of 0.96, an offset of -3.06nA and a perfect correlation coefficient of 1.0000. For constant potential amperometric experiment, it is demonstrated that our amperometric analyzer has comparable detection accuracy and precision with a standard benchtop electrochemical workstation. Moreover, current sensing level A, B and C of our amperometric analyzer were used to obtain testing result shown in Figure 3.11. Measurement accuracy across current sensing levels is therefore demonstrated with this set of data.

3.5 Conclusion

A novel potentiostat circuit, which enables highly accurate bipolar potential control and ultra-sensitive wide range current sensing, has been developed using off-the-shelf IC components under low power and voltage constrains. Theoretical design process in conjunction with systematic performance verification demonstrated excellent performance of this 3-electrode potentiostat. The system has an ultra-low LoD of 5pA with enhanced resolution and accuracy when compared to commercially available mini-potentiostat systems, such as PamSens and WaveNano which have higher limits of detection and lower accuracy. Electrochemical analyses were also conducted using the potentiostat over three gain level settings. Results obtained with the potentiostat show excellent agreement with results obtained using the BAS Epsilon, which again confirm the performance of the system.

In this study, only constant potential amperometry has been implemented. Further development of the system, including expansion of embedded software to conduct other types of amperometric analyses, modification of embedded hardware for increased bandwidth requirement, can be easily performed on top of the current system design. In addition, the high resolution of the amperometric analyzer will make it a suitable system for electrochemical detection in certain biomedical applications, such as for single cell biological analysis with nanoelectrode systems [27-29], real-time extracellular neurochemical detection using carbon fiber microelectrode[30, 31], in which sub-pA level resolution is highly desirable.

REFERENCES

1. Stetter, J.R. and J. Li, *Amperometric Gas SensorsA Review.* Chemical Reviews, 2008. **108**(2): p. 352-366.

2. Cao, Z., W.J. Buttner, and J.R. Stetter, *The properties and applications of amperometric gas sensors.* Electroanalysis, 1992. **4**(3): p. 253-266.

3. Park, J.-K., H.-J. Yee, and S.-T. Kim, *Amperometric biosensor for determination of ethanol vapor.* Biosensors and Bioelectronics, 1995. **10**(6-7): p. 587-594.

4. Romero, M.R., et al., *Amperometric Biosensor for Direct Blood Lactate Detection.* Analytical Chemistry, 2010. **82**(13): p. 5568-5572.

5. Frew, J.E., et al., *Amperometric biosensor for the rapid determination of salicylate in whole blood.* Analytica Chimica Acta, 1989. **224**: p. 39-46.

6. Liu, M., et al., *A disposable amperometric sensor for rapid detection of serotonin in the blood and brain of the depressed mice based on Nafion membrane-coated colloidal gold screen-printed electrode.* Journal of Electroanalytical Chemistry, 2010. **640**(1-2): p. 1-7.

7. Thuerlemann, C., A. Haeberli, and L. Alberio, *Monitoring Thrombin Generation by Electrochemistry: Development of an Amperometric Biosensor Screening Test for Plasma and Whole Blood.* Clinical Chemistry, 2009. **55**(3): p. 505-512.

8. Kwakye, S. and A. Baeumner, *An embedded system for portable electrochemical detection.* Sensors and Actuators B: Chemical, 2007. **123**(1): p. 336-343.

9. Heller, A. and B. Feldman, *Electrochemical Glucose Sensors and Their Applications in Diabetes Management.* Chemical Reviews, 2008. **108**(7): p. 2482-2505.

10. Magjarevic, R., et al., *Amperometric Hydrogen Peroxide Sensors with Multivalent Metal Oxide-Modified Electrodes for Biomedical Analysis*, in *13th International Conference on Biomedical Engineering*, C.T. Lim and J.C.H. Goh, Editors. 2009, Springer Berlin Heidelberg. p. 829-833.

11. Kissinger, P.T. and W.R. Heineman, eds. *Laboratory Techniques in Electroanalytical Chemistry*. 1984, Marcel Dekker, Inc.: New York.

12. Bard, A.J. and L.R. Faulkner, *Electrochemical methods: fundamentals and applications*. 2nd ed. 2001, New York: John Wiley & Sons, Inc.

13. Roe, D.K., *Analytical electrochemistry: theory and instrumentation of dynamic techniques.* Analytical Chemistry, 1978. **50**(5): p. 9R-16R.

14. Ahmadi, M.M. and G.A. Jullien, *Current-Mirror-Based Potentiostats for Three-Electrode Amperometric Electrochemical Sensors.* Circuits and Systems I: Regular Papers, IEEE Transactions on, 2009. **56**(7): p. 1339-1348.

15. Greef, R., *Instruments for use in electrode process research.* Journal of Physics E: Scientific Instruments, 1978. **11**(1): p. 1.

16. *MAX407 1.2uA Max, Single/Dual/Quad, Single-Supply Op Amps Datasheet*, MAXIM.

17. Sodgers, R.S., *Stalking The Wild Potentiostat*, in *Today's Chemist At Work*. 1995, American Chemical Society.

18. Mehta, A., *Understand low-side vs. high-side current sensing*, in *EE Times*. 2009.

19. *INA121 FET-Input, Low Power Instrumentation Amplifier Datasheet*. 1997, Burr-Brown Corporation.

20. Pease, R.A., ed. *Analog Circuits*. 2008, Elsevier INc. : Oxford.

21. *MSp430x4xx Family User's Guide*. 2010, Texas Instruments.

22. *MSP430F47x Mixed Signal Microcontroller Datasheet*. 2008, Texas Instruments.

23. Jin, Y., et al., *Reliable Remote-Monitoring Electrochemical Potentiostat for Glucose Measurements.* Tsinghua Science & Technology, 2009. **14**(5): p. 593-600.

24. Duff, M., *Five basic mistakes to avoid when using instrumentation amplifiers*, in *EE times*. 2007, EE Times Group.

25. Kitchin, C. and L. Counts (2004) *A designer's guide to instrumentation amplifier*.

26. Aamir, M. and M.F. Saleen, *Analysis of noise reduction techniques in embedded systems*, in *National Conference on Emerging Technologies*. 2004.

27. Wu, W.-Z., et al., *Monitoring Dopamine Release from Single Living Vesicles with Nanoelectrodes.* Journal of the American Chemical Society, 2005. **127**(25): p. 8914-8915.

28. Yeh, J.I. and H. Shi, *Nanoelectrodes for biological measurements.* Wiley Interdisciplinary Reviews: Nanomedicine and Nanobiotechnology, 2010. **2**(2): p. 176-188.

29. Zhang, X., et al., *Glucose Nanosensor Based on Prussian-Blue Modified Carbon-Fiber Cone Nanoelectrode and an Integrated Reference Electrode.* Electroanalysis, 1999. **11**(13): p. 945-949.

30. Huang, L., et al., *Detection of exocytosis at individual pancreatic beta cells by amperometry at a chemically modified microelectrode*.

31. Gnegy, M.E., et al., *Intracellular Ca2+ Regulates Amphetamine-Induced Dopamine Efflux and Currents Mediated by the Human Dopamine Transporter.* Molecular Pharmacology, 2004. **66**(1): p. 137-143.

Chapter 4 Additional System Design Considerations

4.1. Low-power mixed-signal circuit design

4.1.1. Core microcontroller module

With smart integrated peripherals and unmatched power efficiency, the MSP430FG479 microcontroller is designed specifically for battery-powered measurement applications. As shown in Figure 4.1, 48 out of the 80 pins in the MSP430FG479 are I/O pins, which can be either used for general-purpose I/O function or configured to connect to a more specialized peripheral. In addition to the I/O pins, the remaining 32 terminals are reserved for specialized tasks such as digital/analog supply voltage, internal reference voltage output, crystal oscillator connection, JTAG interface and etc. Those pins that are not used in this design are terminated properly according to the instruction in the user's guide [1] to avoid additional current consumption and unwanted interference signals.

Every MSP430 flash-based microcontroller implements an embedded emulation module, which support in-circuit programming via low-cost JTAG. A 4-wire JTAG is implemented in this design with a 14-pin male header (Figure 4.1), which can be connected to TI's MSP430 USB-Debug-Interface (MSP-FET430UIF) during programming and debugging.

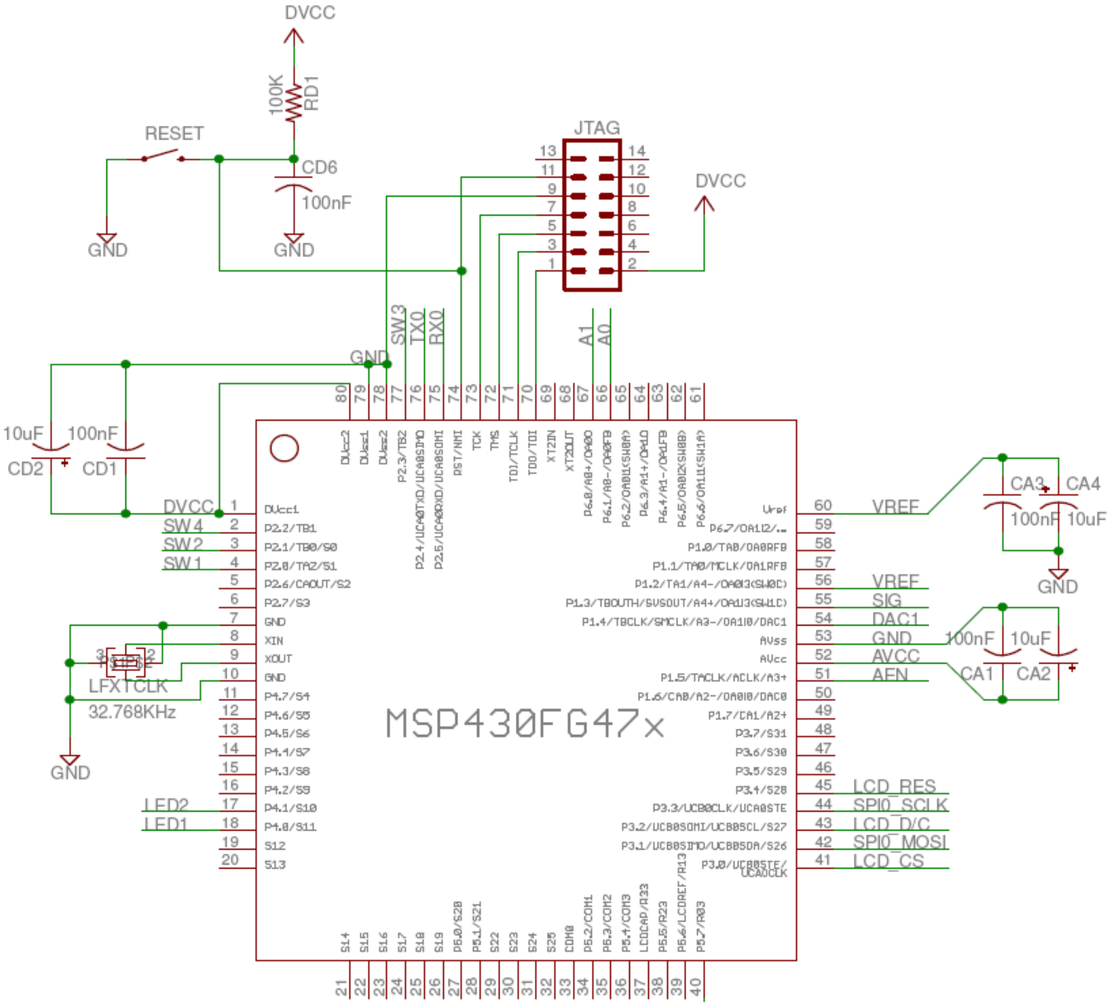


Figure 4.1: Circuit schematic of the core microcontroller module.

The 48 I/O pins are grouped into 6 ports, specifically P1 through P6. Among the 6 ports, P1 and P6 are used to interface with the analog potentiostat circuitry, since the microcontroller's integrated analog peripherals, such as DAC12 and SD16_A, are located on these 2 ports.

The remaining ports, P2 to P5, are used to interact with digital part of the embedded system. To take advantage of the interrupt capability of port 2, four P2 pins are connected to 4 push buttons (SW1, SW2, SW3, SW4) to allow for easy control of the amperometric analyzer. In addition, one of the two universal serial communication interfaces (USCI) is implemented on port 2. The two-pin interface (TX0 and RX0), connected to the USB communication module, is used to transmit and receive data to/from a PC. The other USCI module of the microcontroller locates on port 3. It supports the synchronous peripheral interface (SPI) mode, which is used to control the LCD in our design. Pin 0 and 1 (LED1 and LED2) on Port 4 are configured to control the status of 2 LEDs. And port 5 is unused.

4.1.2. Power supplies

To achieve improved energy efficiency while eliminating power supply noise, a 3V lithium coin cell battery is directly connected to the amperometric analyzer as the major power source. Except for the USB communication module, which is self-powered through a USB cable when active, the amperometric analyzer can operate from a battery potential of 2.5 to 3.0V without sacrificing its performance. The built-in supply voltage supervisor (SVS) indicates a low battery warning when the AVcc supply voltage drops below 2.8V.

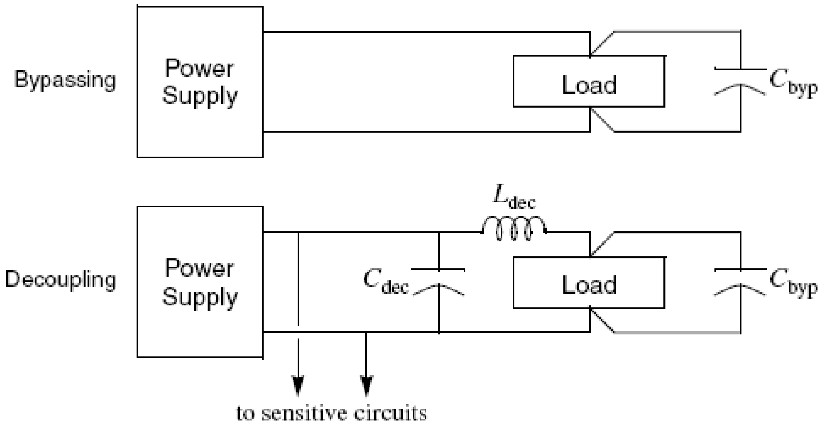


Figure 4.2: A schematic bypassing and decoupling[2].

As shown in Figure 4.2, bypassing and decoupling are two very useful techniques for supply voltage noise reduction[2]. These are used extensively in this design to prevent voltage spikes and high frequency noise in supply rails from corrupt analog performances. Though usually omitted in schematics of published papers and books, decoupling and bypass capacitors are of crucial importance for proper function of the system, especially in a mixed-signal circuit.

A bypass capacitor is required by each analog or digital IC chip. The value of bypass capacitors are usually 100nF, they are required to have short leads to power supply pins, and are placed as close to the devices as possible. It provides a low impedance path to ground for high frequency signals on the power supply. Especially in the analog layout design, proper bypassing is of crucial importance not only in terms of reducing noise, but also it can prevent system failure due to oscillation caused by out-of-band power supply noises. In our design, at least one bypass capacitor is used for each IC according to the recommendation in manufacturer's datasheet.

Used for power supply noise reduction as well, decoupling is however a distinct concept. As shown in Figure 4.2, a decoupling circuit serves as a low pass filter and is used to isolate two circuits on a common line[2, 3]. Specifically in our system, ripple and noise spikes on the power supply line due to digital switching can corrupt the conversion result of SD16_A, so decoupling capacitors are placed between DVcc, AVcc, Vref to ground according to the user's guide(Figure 4.1)[1]. These decoupling capacitors serve as mini charge reservoirs which can provide additional charge when needed, and is used to improve the power supply stability. To achieve high conversion accuracy, an additional decoupling circuit is designed to prevent transmission of supply voltage noise from the digital circuitry to analog circuitry from the source (Figure 4.3)[3]. Therefore, whenever a transient voltage spike occurs in the digital power rail, electrical charge stored in the decoupling capacitor is released to supply the power need without disturbing the stability of analog power.

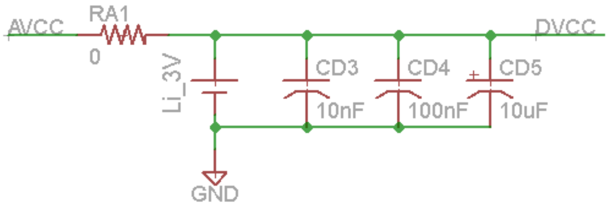


Figure 4.3: Schematic of the filter circuit used to isolate noisy digital power rail (DVcc) from a clean analog power supply (AVcc).

4.1.2. Digital circuitry

The digital circuitry of the amperometric analyzer contains mainly the USB communication module and the user interface module.

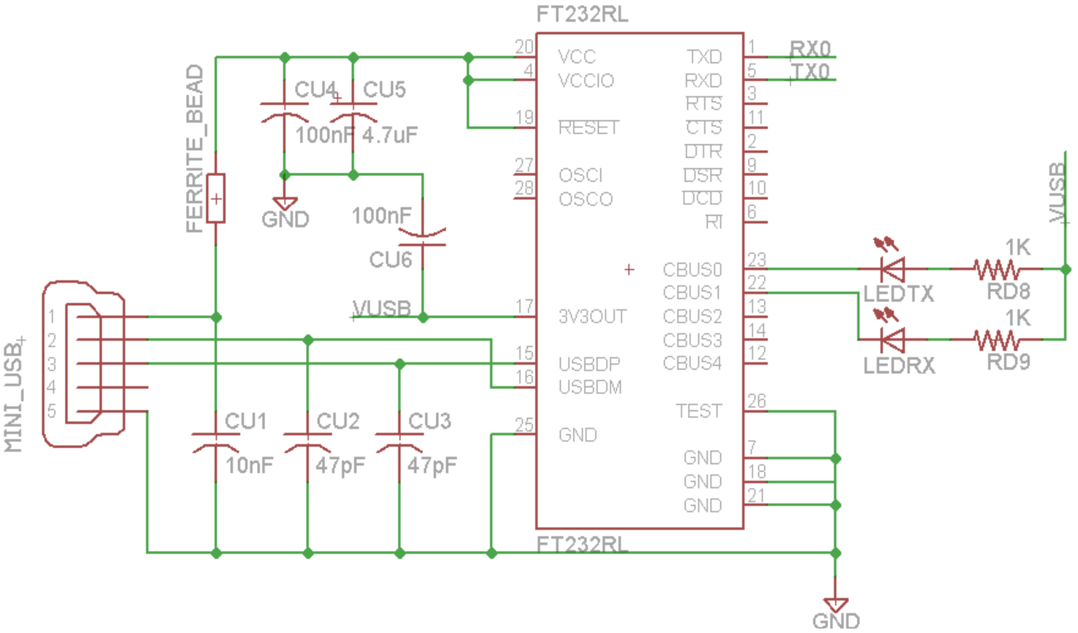


Figure 4.4: Schematic of the USB communication module.

As shown in Figure 4.4, to support USB communication with the 2-wire asynchronous UART interface of the MCU, a FT232RL chip from Future Technology Devices International (FTDI) is used. The FT232RL chip provides a fully integrated solution for communication between the embedded system and the user's PC with minimum external components and board space. In order to function properly, the RX and TX pin of the MCU needs to be connected to the TX and RX pin of the FT232RL chip respectively (Figure 4.4).

A NOKIA 3310 graphic LCD, 2 status LED and 4 push buttons constitute the user interface of the system (Figure 4.5). Two LED are used to signal power up of the system as well as successful completion of an experiment. The LED is off when the control signal is high, and can be turned on by toggling the P4.0 and P4.1 output to low. Four pushbuttons are connected to the microcontroller on pins 0 to 3 of port 2. When a button is pressed, an interrupt is generated to initiate corresponding actions. The NOKIA3310 LCD is a low power serial LCD that has an 84 by 48 pixel resolution. It uses the Philips PCD8544 controller, and is controlled via a simple SPI protocol.

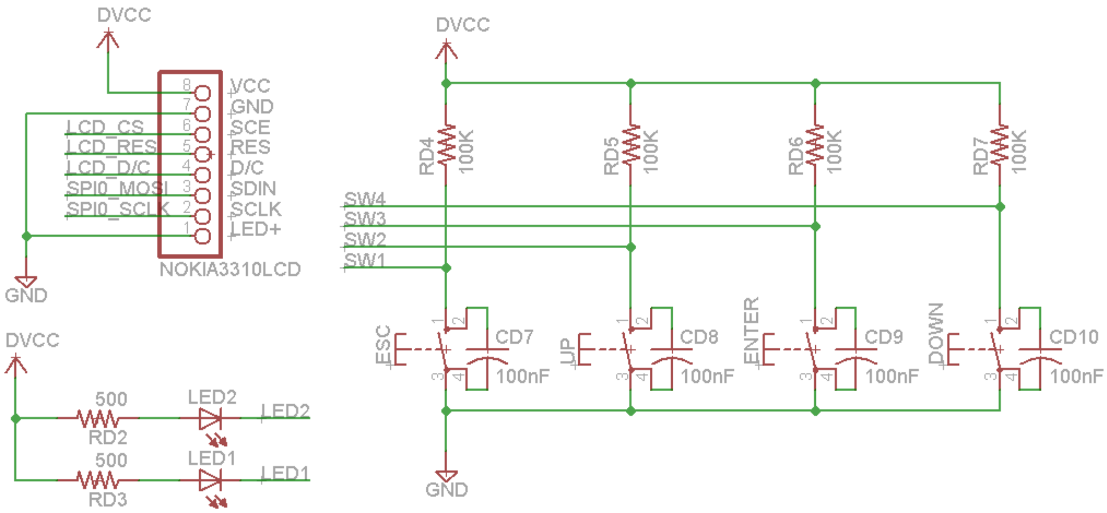


Figure 4.5: Schematic of the user interface module. It consists of a graphic NOKIA3310 LCD, 2 status LED and 4 push buttons.

4.1.3. Analog circuitry

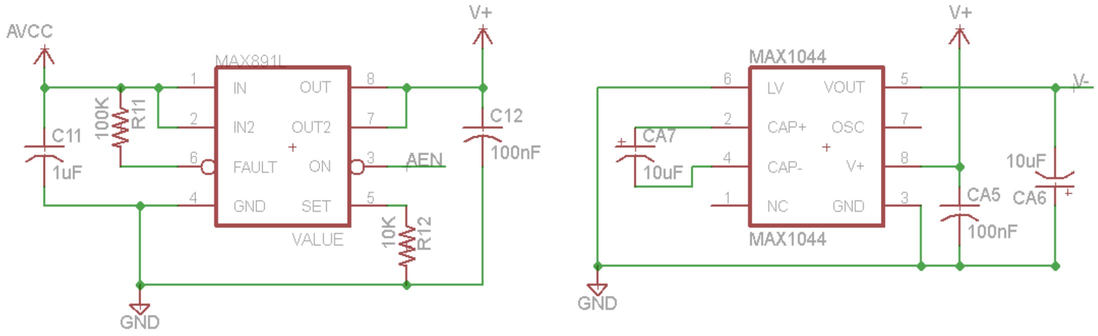


Figure 4.6: Schematic of the analog power switch (left) and the negative voltage generator (right).

The analog potentiostat of the amperometric analyzer contains an op-amp circuit that conditions redox potential and the differential amplifier circuits that measure the redox current. Analog circuitry has been discussed in detail in chapter 2 and 3, except for the analog power switch and the negative voltage converter circuit (Figure 4.6).

The power supply to the entire analog circuitry (V+ in Figure 4.6) is controlled using a MAX891L low-voltage high-side switch. The control signal (AEN) is active low. Therefore, when AEN is logic high, the MAX891L is switched off which causes the output voltage drop to ground level.

In order to conduct bidirectional potential control and current sensing, a negative power supply is required by op-amps. The differential amplifier A MAX1044 switched-capacitor voltage converter is therefore used to invert the +3V supply voltage to -3V. This voltage converter circuit is placed after the analog switch, so it will turned off together with other parts of the analog circuitry when not in use.

4.2. Two-layer PCB for precision analog signal sensing

4.2.1. Device placement on the mixed-signal board

The layout requirements become more stringent with increasing analog-to-digital conversion resolution[4]. The first step towards implementation of a precision 16-bit circuit layout is to partition the devices on board according to their noise related characteristics.

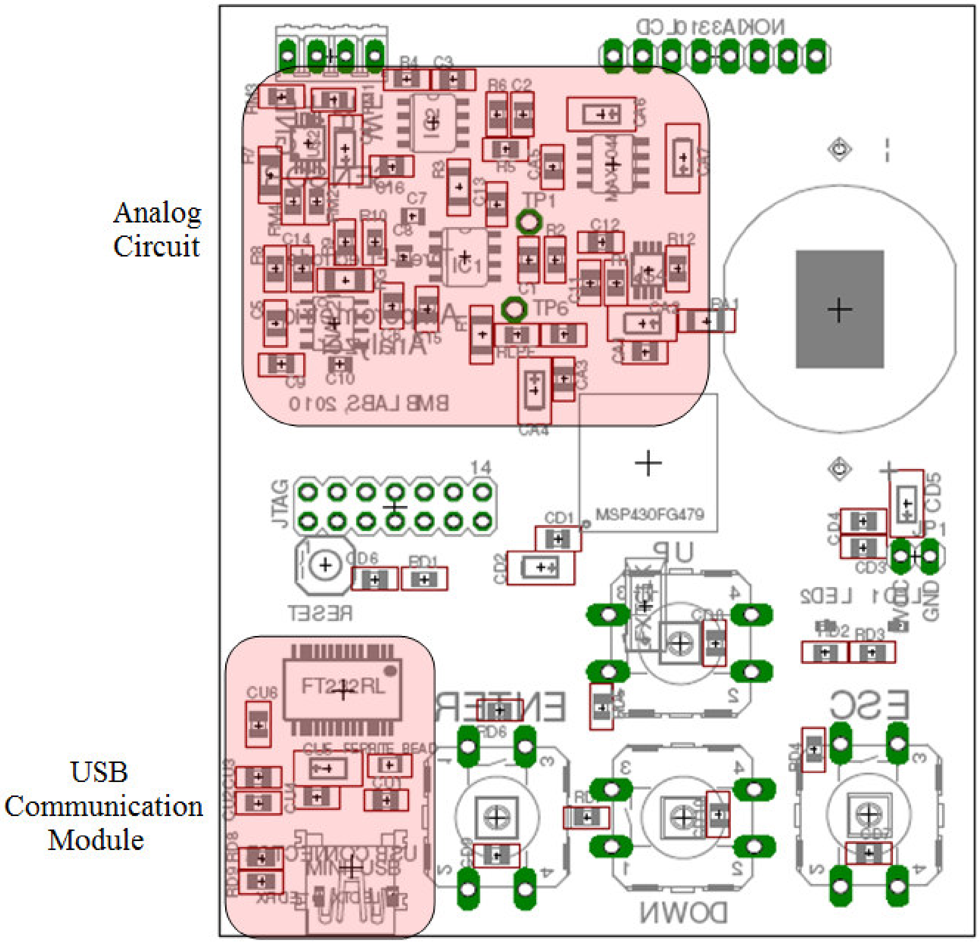


Figure 4.7: Device placement of the custom PCB board for the amperometric analyzer.

Noise creator devices are usually high-speed digital in nature, which in our case is the USB communication module. A good board floor planning requires maximum separation of high-speed digital devices from analog components, thus to prevent high-speed logic from polluting analog signal paths. As shown in Figure 4.7, the FT232RL communication chip is located at the bottom-left corner of the PCB. With analog interfacing pins of the MSP430FG479 microcontroller located at the upper-left corner of the device, the noise-sensitive analog potentiostat circuit is placed at the upper-left corner of the PCB to maintain minimum interconnect trace lengths. By physically separating the sensitive analog device from noise creator devices, this design can effectively reduce switching noise pickup due to crosstalk and EMI[4].

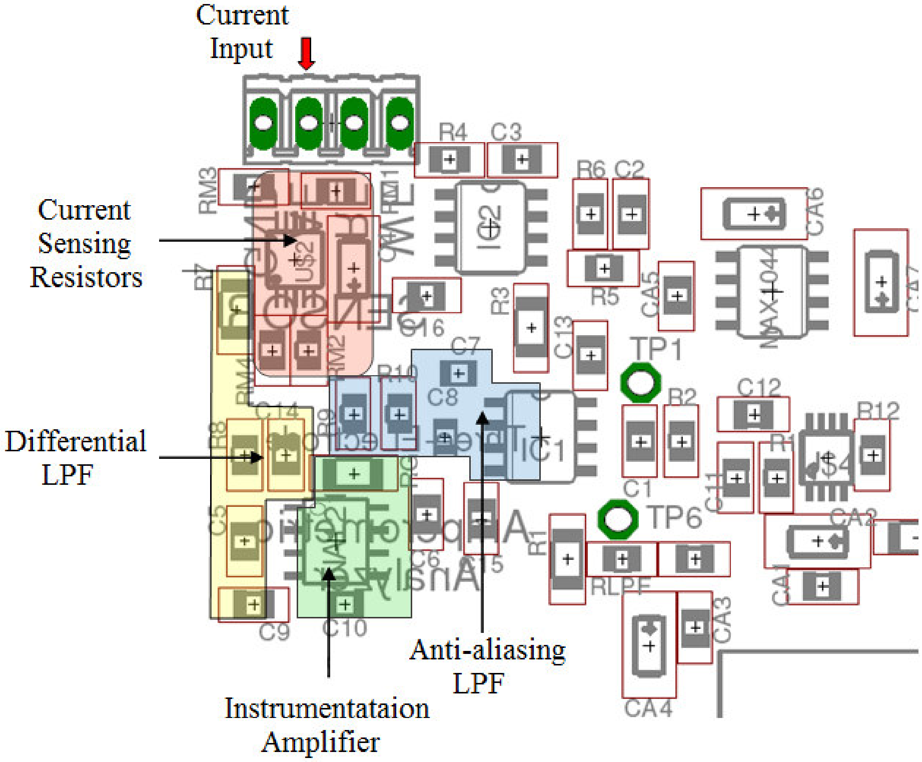


Figure 4.8: Zoom-in view of device placement inside the analog circuit block.

Placement of devices within the analog circuit block is also very important, especially for the current sensing circuitry. Current signals in picoampere range, even if only traveled for a short distance on the PCB board, will quickly be buried into noises and become impossible to measure. After the sensor current signal enters the PCB, it must be immediately converted into a differential voltage signal, filtered and then enter the instrumentation amplifier (Figure 4.8). With a voltage gain of 100, the instrumentation amplification converts the initial signal into a much larger voltage signal, which has 100 times better immunity to noises existing on the PCB board.

The A/D converter in the MSP430FG479 has a least significant bit (LSB) of $1.2\text{V} \div 2^{16} = 18.3\mu\text{V}$, which can be affected by any error in the layout[5]. Therefore, design effort was focused on ensuring good ground connections and signal trace routing, which are critical in precision 16-bit circuit.

4.2.2. Power supply layout strategy

Generally speaking, there exist three types of power-distribution system (Figure 4.9). In the single-point system, each active component has a separate power trace until all met at a single source point (Figure 4.9a). In the contrary, in a multipoint system, the connections to the power supply are made one after another, and are thus prone to common impedance coupling (Figure 4.9c). When implementing a single-point system is not feasible, star topology can be used as an alternative method for power connection (Figure 4.9b).

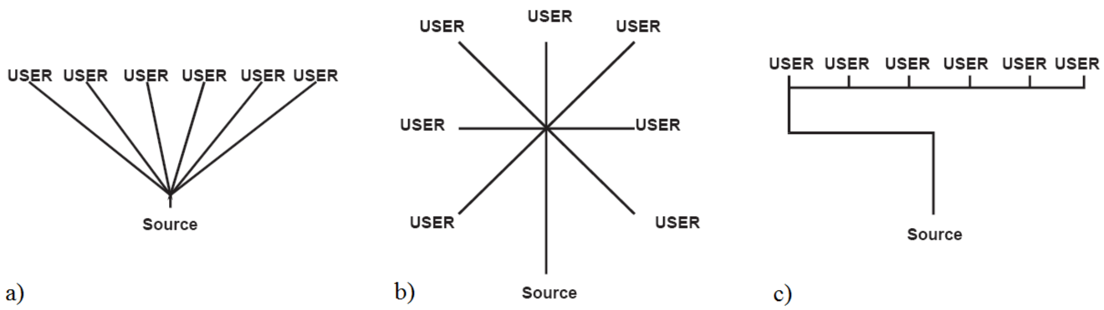


Figure 4.9: Schematic of three power distribution systems. (a) is the single point system, (b) is the start connection, and (c) is the multi-point connection method.

The final power supply layout strategy adapted in this design is a modified star connection method. Specifically, two star connections are implemented in the analog circuitry and digital circuitry separately. Plus the decoupling circuit between the analog and digital power (Figure 4.3), crosstalk between digital and analog power rails, as well as between power supplies of any two digital/analog devices can be mostly eliminated.

4.2.3. Ground plane and signal trace routing

Although a ground signal appears to be a simple design on a schematic, the performance of a mixed-signal embedded system is largely dictated by the layout of ground traces.

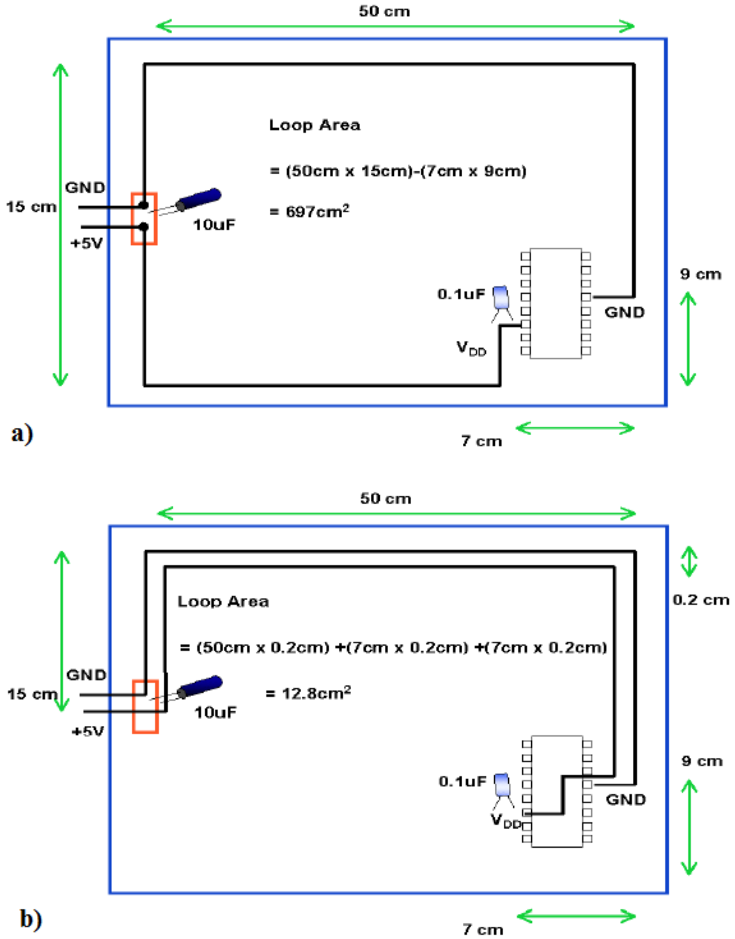


Figure 4.10: The power and the ground traces layout in (a) creates a 697 cm^2 current loop, which opens the opportunity for EMI. With better matched power and ground traces, the loop area and thus EMI opportunity is decreased by 54 times [4].

Current flow through a system via any power or signal connection must return to the power supply though ground, thereby completing the circuit[6]. The loop area enclosed by the power and ground traces caused by improper design is a major cause of noise and interference[7]. As shown in Figure 4.10(a), the 697 cm^2 current loop forms an antenna, which can radiate as well as pick up energy. Therefore, a minimized current loop area is always desirable, as it can reduce system noise level caused by EMI (Figure 4.10(b)) [4]. A ground plane is an effective method to keep all outgoing signals and return paths as close together as possible. With a solid ground plane, the return signal can always follow the lowest impedance path, which is directly under the signal trace[7]. It is often implemented as an internal layer in a multi-layer PCB to minimize EMI problems and possibly eliminate the need of costly shielding[4].

In our design, the bottom layer of the 2-layer PCB is reserved for the ground plane and the top layer is mainly used for surface mount device connections and signal trace routing. Ground fill was used on the open area of the top layer, with the USB connection module block being an exception. Several vias are used to connect the top layer ground fill to the bottom layer ground plane to minimize impedance between them.

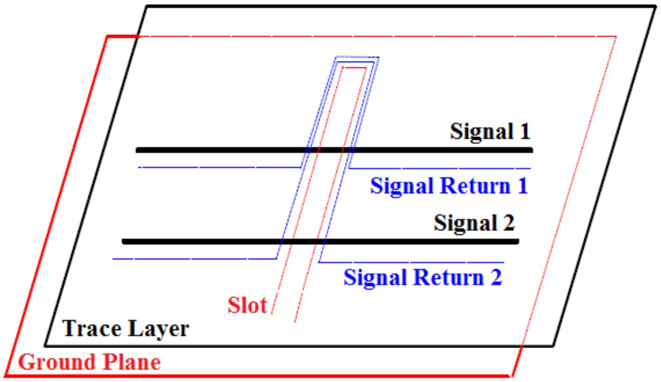


Figure 4.11: Slot on ground plane can cause EMI radiation and crosstalk problems [8].

However, due to the circuit topology, it is inevitable to have connections on the bottom layer, which create slots in the ground plane. These increase the current loop area due to the redirection of the return current (Figure 4.11) [8]. In addition, slots in the ground plane also significantly increase the crosstalk between the trace that crosses over them[4, 8]. Added precautions were taken during the layout process to minimize slots on the bottom layer, especially within the analog circuit block. As shown in Figure 4.11, the majority of connections were implemented on the top layer, which left only a small number of slots with minimum length on the bottom ground plane.

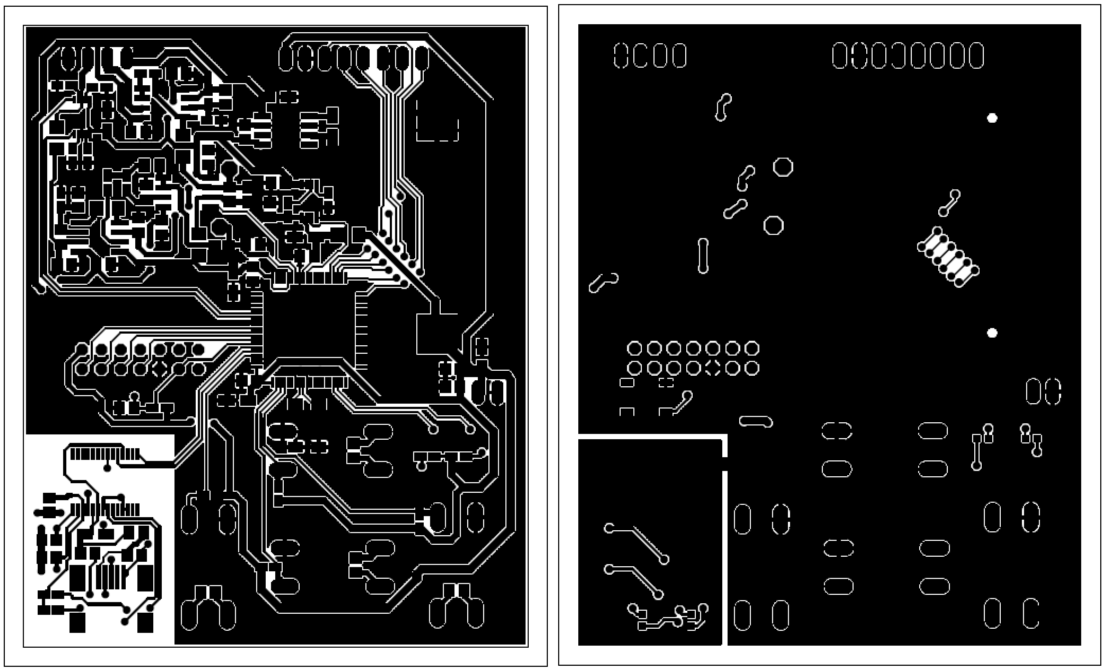


Figure 4.12: Top copper (left) and bottom copper (right) layer of the system PCB layout.

In Figure 4.13 the final PCB layout is shown. Analog and digital signal traces are strictly separated. To minimize copper resistance and thus preserve signal integrity, wide traces were used for power and important signal connections. The potentiostat block was laid out using minimum distance track routing, with all devices placed as compact as possible. In order to minimize the number of bottom layer signal connections, several resistors (RG, R1, R3, R7) with larger package size were used, as they allow for signal traces going between their connector pads, therefore eliminating the need for these signal traces to go through the bottom layer. With the current layout design, amperometric current detection with a 5pA detection limit and sub-pA measurement resolution can be achieved without external shielding.

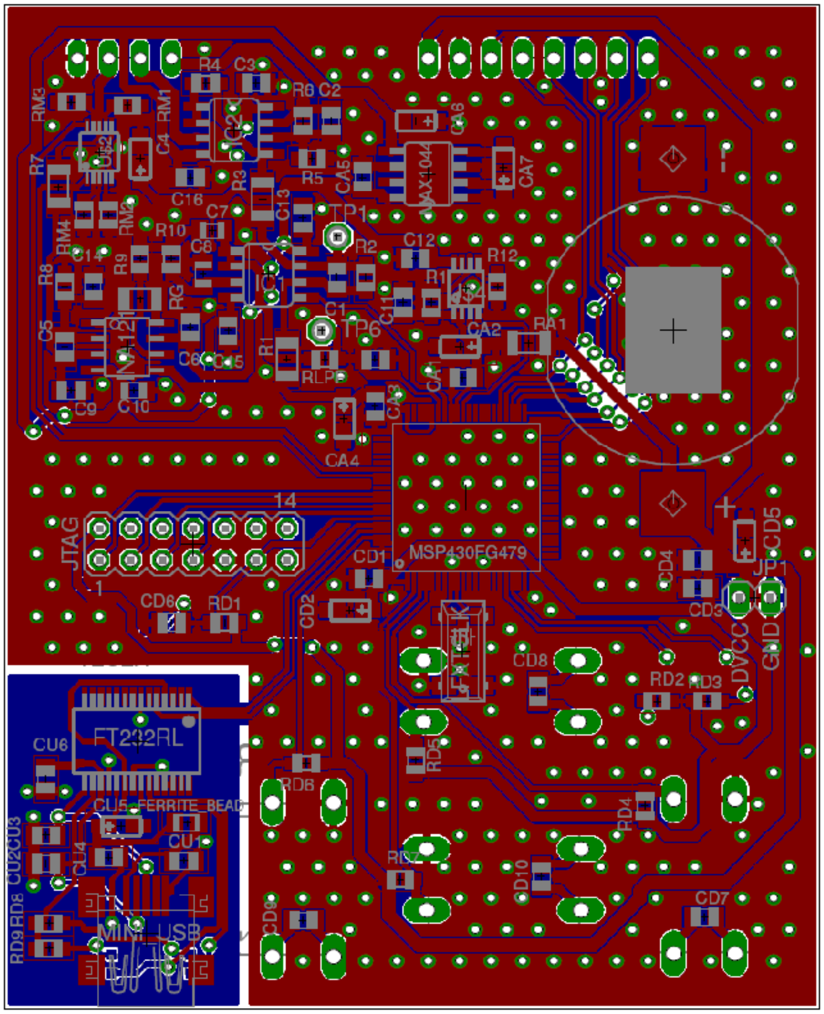


Figure 4.13: PCB layout of the amperometric analyzer. Red represent top layer and blue represent bottom layer.

REFERENCES

1. *MSp430x4xx Family User's Guide*. 2010, Texas Instruments.
2. Kundert, K. (2004) *Power Supply Noise Reduction*. 12.
3. Mustafa, K. (2001) *Filtering Techniques: Isolating Analog and Digital Power Supplies in TI's PLL-Based CDC Devices*. 7.
4. Baker, B.C. *Analog and Interface Guide*. **1**.
5. *PCB layout tips for high resolution*, in *Precision Analog Applications Seminar*, Texas Instruments.
6. Williams, T. (2007) *The ground plane: lord of the board.*
7. Gray, N. (2006) *An intuitive, practical approach to mixed-signal grounding*. EE Times.
8. Brooks, D. (1999) *Slots in Planes*. Printed Circuit Design.

Chapter 5 Conclusion and future improvements

This thesis has presented a novel miniaturized amperometric analyzer for field appropriate monitoring applications. Using carefully selected commercial-off-the-shelf components, the embedded potentiostat possesses a unique combination of advantages including small self-contained package with ultra-low power consumption, accurate bidirectional redox potential control from -1.2V to +1.2V, high resolution current detection from ±5pA to ±1mA covered by 4 software selectable detection levels, improved stability unaffected by electrochemical sensor impedance, and high EMI immunity without external shielding.

The superior performance of the system is achieved with combined design efforts of robust embedded hardware/software design, careful IC selection, effective signal filtering techniques and good PCB layout practice. In this thesis, important design tradeoffs, rationales behind various decisions and detailed theoretical calculations were presented, followed by a systematic evaluation of the device prototype. The obtained device specification successfully meets all preset design goals. Such a system not only provides scientists with a powerful research tool, but will also, in the future, expand the pool of prospective applications and improve the user experience of future customers.

Future improvements of the system can take place in several design aspects.

First and foremost, in order to moving this design from the lab to the market for mass production, a quick and simple calibration protocol need to be established to account for device to device variation. A system calibration is required before the first use of each device. This is especially important for the sensitive current detection block, which requires calibration to eliminate system offset and to obtain individual current to voltage transfer coefficient for each current detection level.

An optional wireless control and data transmission feature can be easily added to the current system, which allows remote control and real-time data analysis using personal computer. This additional feature can potentially provide an excellent solution for remote environmental monitoring, or for real-time amperometric detection inside a controlled environment, which is typically encountered in chemical and pharmaceutical industry.

To further expand the application range of the amperometric analyzer, experimental protocols for other amperometric techniques can be added in the embedded software to allow multitude detection functions. In addition to constant potential amperometry, other popular techniques such as cyclic voltammetry, square wave and pulse voltammetry, which require more complex redox potential waveforms, can be made available to user in upgrade software packages. Since the signal filtering circuit in the current device is designed for 1Hz sample rate, this block of the hardware may require modification to meet new bandwidth requirement. If required, software selectable low pass filters with different cutoff frequencies can also be implemented to accommodate bandwidth and sensitivity tradeoff in the wide current range potentiostat.

A huge array of sensing technologies, based on a wide range of chemical or biological recognition mechanism, has been developed utilizing amperometric detection principle. Designed to serve as a generic detection system for different electrochemical sensors, this basic platform can also be customized for specific application. For example, this mini-potentiostat can be incorporate into micro-total analysis systems for electrochemical detection of biochemical assays in hospital point-of-care settings or by caregivers in non-hospital settings. With a sensor loading dock, after insertion of target sensor cartridge, the fully automated system will allow immediate amperometric analysis according to preset protocols. The system can also be programmed to display analyte concentrations instead of current readings. This type of customized system is expected to provide the user with improved detection efficiency due to minimal manual setting procedures and less noise intervention due to cable-free connection. Furthermore, the system can be easily configured to detect multiple clinical significant analytes if different electrochemical sensors are available in unified connector and package sizes. Similar customized design can be potentially useful in environmental monitoring sectors, food and agriculture industries.

In this design, the lower current measurement range has been pushed down to reach the theoretical limit in a mixed-signal embedded system. So in future customized systems, current measurement range can be easily tailored according to specific detection needs, as long as it falls into the 5pA to1mA range.

APPENDIX

A.1 Main Module for Amperometric Analyzer

```
#include <msp430fg479.h>

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include "pcd8544.h"          //LCD code


//hardware

#define LED1  BIT0

#define LED2  BIT1

#define B1  (1<<0)

#define B2  (1<<1)

#define B3  (1<<2)

#define B4  (1<<3)


//constant definitions

#define ON 1

#define OFF 0

#define RX_MSG_MAX 32              //maximum receive buffer size

#define TX_MSG_MAX 32              //maximum transmit buffr size

#define CR 0x0D

#define LF 0x0A


//memory definition
```

```
#define DATA_START      0x0200            //start of data memory
#define DATA_END  0xFDFF            //end of data memory
#define INFO_A          0x1080            //start of segment A in
info memory
#define INFO_B          0x1000            //start of segment B in
info memory

#define YEAR1           2000
#define LEAP(year)      (!((year)%4) && (((year)%100)
|| !((year)%400)))

//text messages
static char introMsg[]={"\n\rMiniEC Sensor Monitor Online ... \n\r"};
static char helpMsg[]={"\n\rEC Sensor Commands Menu:\
                                                \r(hp)Help\
                                                \r(gs)Get Status\
                                                \r(gf)Get Files\
                                                \r(sc)Clear Files\
                                                \r(sp ival dur
pot)Set Parameters\
                                                \r(gt)Get Time\
                                                \r(st day-mon-yr
hour:min)Set Time\
                                                \r(mr)Start
Measurement\
                                                \r(ms)Stop
Measurement\n\r"};

//time variables
unsigned int interval = 1;          //interval in seconds
unsigned long ntimes = 1;          //interval interval-timer units(s)
unsigned long nticks = 0;           //interval counter
unsigned int duration = 1;          //duration in minutes
unsigned long dtimes = 0;     //duration in interval-timer units(s)
unsigned long dticks = 0;           //duration counter
unsigned int dsec = 0;
unsigned int dmin = 0;
unsigned int offset = 0;
unsigned int initial = 0;
unsigned int output;
unsigned int range = 0x03;

void InitializeSystem(void);

//analog and digital convertors
void DAC_init(void);
void DAC_stop(void);
void DAC_start(int potential);
void ADC_init(void);
void ADC_read(void);
void ADC_stop(void);
void end_measure(void);

//usart0
void usart0_init(void);
void usart0_send_char(char c);
void usart0_send_string(char msg[]);
void usart0_send_int(int value);
void usart0_rx_isr_enable(void);
void usart0_rx_isr_disable(void);

//timer
void start_interval_timer(int ival, int dur);
void stop_interval_timer(void);

void delay_ms(unsigned int);

//time structure
typedef struct{
      int hours, mins, secs;
      int days, month, year;
}rtcTime;

rtcTime startTime;
rtcTime stopTime;
rtcTime currentTime = {12, 0, 0, 24, 8, 2010};        //current date
and time
```

```
unsigned char *month_name[] = {"Jan", "Feb", "Mar", "Apr", "May",
"Jun",
                                                "Jul", "Aug", "Sep",
"Oct", "Nov", "Dec"};

//****FSM definitions and variables***//
#define NUMBER_EVENTS 7
#define NUMBER_STATES 6

//????
#define FTIME     0x01
#define FIVAL     0x02
#define FB1       0x04
#define FB2       0x08
#define FB3       0x10
#define FB4       0x20
#define FDONE     0x40

//events
#define NONE 0
#define B1KEY 1
#define B3KEY 2
#define B2KEY 3
#define B4KEY 4
#define TIMEOUT 5
#define IVAL 6
```

```
//states
#define IDLE 0                  //display time, low power mode 3
#define MENU 1                  //low power mode 0, wakeup to display
menu, waiting for action
#define READY 2               //calibration and ready to take
measurement
#define RECORD 3        //measurements with storage
#define SETUP 4
#define LOGS 5

//variable defined elsewhere
static unsigned int potential = 300;                    //400mV bias
potential
static unsigned int dacValue = 0x400;                   //300mV digital
value
//static unsigned int dacValue = 0x555;                 //400mV digital
value
static unsigned int result_1Hz;
static unsigned int initial_result;
static unsigned int index = 0;
static unsigned int duplicate = 8;
static char usart0_rx_buffer[RX_MSG_MAX];       //USART receive
buffer
static char usart0_tx_buffer[TX_MSG_MAX];       //USART transmit
buffer
```

```
static char buffer[16];

unsigned short usart0_rx_flag = OFF;      //indicates received serial
dta
//unsigned short monitor_flag = OFF;            //indicates
presence/absence of PC
unsigned char usart0_rx_index = 0;

//flash memory variables
unsigned int *data+pointer = (unsigned int *)DATA_START;    //Start
address of data in next file
unsigned char num_files = 0;        //number of files saved in memory
//memory available
unsigned int mem_total = (unsigned int)DATA_END - (unsigned
int)DATA_START;
unsigned int mem_left = (unsigned int)DATA_END - (unsigned
int)DATA_START;

//flash memory functions
void flash_write(void *dst, const void*src, unsigned int size);
void flash_write_word(void *dst, const void *src, unsigned int size);
void flash_erase_segment(void *address);
void flash_clar_memory(void);
void write_SegA(char value);
void logHeader(int mark1, int mark2);
void logData(int mydata);
```

```c
void logTail(int marker);
int * find_next_location(void);

//type definitions
//file header
typedef struct{
	unsigned int marker1;
	unsigned int fileID, potential, interval, duration;
	unsigned int year, month, day;
	unsigned int hour, min, sec;
	unsigned int marker2;
}file_header;

//file footer
typedef struct{
	unsigned int marker;
	unsigned int max, average;
}file_footer;

//state table entry
typedef struct{
	void (*Actions)(void);
	unsigned char state_next;
} state_entry;

//state variable initialization
```

```
static volatile unsigned char event_tray;
static volatile unsigned char one_event;
static unsigned char mode;
static unsigned char mode_prev;
static const state_entry mainStates[NUMBER_EVENTS][NUMBER_STATES];

//prototype functoins
void FSM_init(void);
void FSM_run(void);
unsigned char FSM_get_event(void);


//prototype functions for top level actions
void none(void);
void idle(void);
void wakeup(void);
void ready(void);
void record0(void);
void record(void);
void done(void);
void pause(void);
void setup(void);
void logs(void);
void setup_run(void);
void logs_run(void);
void respond_to_cmd(void);
```

```
//flash memory and data storage

void initFileSystem(void);

void startDataRecord(void);

void saveResult(void);

void Retrieve(void);

void Erase(void);

void eraseSeg(char seg);

void writeByte(char *address, char value);

void writeWord(int *address, int value);

void writeMsg(char *address, char msg[]);

void makeFileEntry(void);

void makeDataHeader(void);

void logHeader(int mark1, int mark2);

void logData(int mydata);

void logTail(int marker);

void transmitData(int mydata);

int *findNextLocation(void);

void rtc_time_string(unsigned char *buffer);


//macros

#define receive_command(a,b) (usart0_rx_buffer[0]==a &&

usart0_rx_buffer[1]==b)

 //general function


//main state table
```

```
extern const state_entry mainStates[NUMBER_EVENTS][NUMBER_STATES]={
      /*  IDLE           MENU          READY           RECORD
SETUP             LOGS        */
      {{none,IDLE},  {none, MENU},  {none,READY},    {none,RECORD},
{none,SETUP},     {none,LOGS}},  //noevent
      {{idle,IDLE},  {idle,IDLE},   {wakeup,MENU},   {pause,READY},
{idle,IDLE},      {idle,IDLE}},//B1 pressed (ESC)
      {{wakeup,MENU},{ready, READY},{record0,RECORD},{none,RECORD},
{setup_run,SETUP},{logs_run,LOGS}},//B3 pressed (ENTER)
      {{none,IDLE},  {setup, SETUP},{none,READY},    {none,RECORD},
{logs,LOGS},      {wakeup,MENU}}, //B2 pressed (UP)
      {{none,IDLE},  {logs, LOGS},  {none,READY},    {none,RECORD},
{wakeup,MENU},    {setup,SETUP}},//B4 pressed (DOWN)
      {{none,IDLE},  {idle,IDLE},   {idle,IDLE},     {done,READY},
{idle,IDLE},      {idle,IDLE}},     //timeout
      {{none,IDLE},  {none,MENU},   {none,READY},
{record,RECORD},{none,SETUP},     {none,LOGS}}  //measurement
interval
};

int main(void){

      //initialize MCU and peripherals
      InitializeSystem();

      usart0_send_string(introMsg);
      //enable all interrupts
      _EINT();

      //find the next location to write data
      datapointer = findNextLocation();

      FSM_init();

      while(1){

            if(usart0_rx_flag == ON){respond_to_cmd();}

            FSM_run();

            if(usart0_rx_flag == ON){respond_to_cmd();}

            LPM3;
      }
 }

//Initializeds the miniEC FSM
//-no events in the event tray
//-current mode is IDLE
//-previous mode is IDLE
void FSM_init(void){
      event_tray = NONE;
      mode = IDLE;
      mode_prev = IDLE;
}

//Fetches an event, executes appropriate action and performs
transition to next mode
void FSM_run(void){
      unsigned char one_event;
      char buffer[32];

      one_event = FSM_get_event();        //get an event

      sprintf(buffer, "event = %d\n\r", one_event);
      usart0_send_string(buffer);

      mainStates[one_event][mode].Actions();    //execute the action
      mode_prev = mode;                         //record current
state
      mode = mainStates[one_event][mode].state_next;  //tansition to
next state

      sprintf(buffer, "next = %d\n\r", mode);
      _send_string(buffer);
}
```

```
//Fetches events in order of priority from event tray
unsigned char FSM_get_event(void){
    //button 1 ESC
    if(event_tray & FB1){
        event_tray &= ~FB1;
        return B1KEY;
    }

    if(event_tray & FDONE){
        event_tray &= ~FDONE;

        usart0_send_int(00);
        return TIMEOUT;
    }
    if(event_tray & FIVAL){
        event_tray &= ~FIVAL;

        return IVAL;

    }
    //button 3 ENTER
    if(event_tray & FB3){
        event_tray &= ~FB3;
        return B3KEY;
    }
```

```
      //button 2 UP
      if(event_tray & FB2){
            event_tray &= ~FB2;
            return B2KEY;
      }

      //button 4 DOWN
      if(event_tray & FB4){
            event_tray &= ~FB4;
            return B4KEY;
      }
      //default event
      return NONE;
}

//Port2 ISR to detect button presses
#pragma vector=PORT2_VECTOR
 __interrupt void port2_isr (void){
      delay_ms(10);     //debounce button for 10ms

      if(P2IFG & B1){
            event_tray |= FB1;
            P2IFG &= ~B1;                                  //clear
B1 interrupts flag
            P4OUT ^= BIT0;
      }
      if(P2IFG & B2){
            event_tray |= FB2;
            P2IFG &= ~B2;                                        //clear
B2 interrupts flag
            P4OUT ^= BIT1;
      }
      if(P2IFG & B3){
            event_tray |= FB3;
            P2IFG &= ~B3;                                        //clear
B3 interrupts flag
            P4OUT ^= BIT0;
      }
       if(P2IFG & B4){
            event_tray |= FB4;
            P2IFG &= ~B4;                                        //clear
B3 interrupts flag
            P4OUT ^= BIT1;
      }
      LPM3_EXIT;
 }

//none
void none(void){}

//idle: diaplay user menu
void idle(void){
      stop_interval_timer();
      usart0_send_string("\n\r#idle\n\r");

      unsigned char ampm = '1';
      unsigned char time_line[14] = "";
      unsigned char data_line[14] = "";
      unsigned char mem_line[14] = "";
      unsigned int mem_time = get_mem_time(interval);
      time_record now = current_time;
      if(now.hour > 12){                  //check for am or pm
            ampm = 'p';
            now.hour -= 12;
      }

      sprintf(time_line, "%02d:%02d%c%c", now.hor, now.min, ampm,
'm');
      sprintf(data_line, "%2d-%3s-%04d", now.day,
month_name[now.month-1],
                  now.year+YEAR1);
      sprintf(mem_line, "%5d Minutes", mem_time);

      lcd_clear();
      lcd_write_string(5,0, "Idle");
      lcd_draw_box(0,0,83,8, PIXEL_XOR);
      lcd_write_string(3,2, time_line);
      lcd_write_string(0,4, mem_line);
```

```
        lcd_update();

}

//wakeup: update LCD to enter measurement mode
void wakeup(void){

        unsigned int timeout = 5;                            //5
minute timeout duration

        lcd_clear();
        //clear the LCD
        lcd_write_string(0,1, "Experiment");
        lcd_write_string(0,3, "Setup");
        lcd_write_string(0,5, "Logs");
        lcd_draw_box(0,0,83,8, PIXEL_XOR);
        lcd_update();

        if(monitor_flag == ON){
                usart0_send_string("\n\r#Wake. Type\'mr\'again to start
experiment\n\r");
        }
        stop_interval_timer();

        start_interval_timer(interval, timeout);
}
```

```
//ready: stop any already running process, check memory
//           calibrate the system, calculat the initial offset
//       starts an interval timer to time out if not action follows
void ready(void){
      unsigned char buffer[64];
      unsigned int infosize = sizeof(file_header) +
sizeof(file_footer);
      unsigned int mem_needed;

      stop_interval_timer();

      //range selection
      P6OUT = range;

      //calibration process
      ADC_init();                              //initialize ADC
      initial_result = 0;

      delay_ms(5000);
      ADC_read();                              //read 1 round
      while(index < duplicate){
            ADC_read();
            initial_result += result_1Hz;
            index++;
      }
      index = 0;
      offset = initial_result>>3;         //get the average of 8
readings

      //determine available memory and how much is needed for data
storage
      //with current interval and duration configuration
      mem_needed = infosize +
(((duration*60)/interval)*(sizeof(int)*5));

      sprintf(buffer, "\n\rdatapointer %d %p %u\n\r",
                  (int)data_pointer, &data_pointer, *data_pointer);
      usart0_send_string(buffer);

      sprintf(buffer, "d=%d, h=%d, t=%d, mn=%d, ml=%d\n",
            sizeof(int), sizeof(file_header), sizeof(file_footer),
mem_needed, mem_left);
      usart0_send_string(buffer);

      result = 0;
      result_avg = 0;
      result_sum = 0;

      lcd_clear();                                    //clear the LCD
      lcd_write_string(5,0, "Ready");
      lcd_draw_box(0,0,83,8, PIXEL_XOR);
```

```
    //if not enough memory, notify user but continue
    if(mem_neede > mem_left){
        lcd_write_string(1,1, "Low memory!");
    }
    lcd_write_string(0,3, "Press -");
    lcd_write_string(0,4, "\"Enter\" to run");
    lcd_write_string(0,5, "\"Stop\"to exit");
    lcd_update();

    stop_interval_timer();

    usart0_send_string("\n\r#Ready. Plug in Sensor. Type \'mr\'
again to start \r\n");
    LED1 |= ON;

    unsigned int timeout = 5;                                   //5
minute timeout duration
    start_interval_timer(interval, timeout);
}

//record0: first recording activity--starts ADC and DAC, notes time
and writes file header
void record0(void){
    unsigned char line1[14] = "";
```

```
      stop_interval_timer();
      DAC_init();                               //initialize DAC

      lcd_clear();
      sprintf(line1, "00:00/%02d:00", duration);
      lcd_write_string(2,0,line1);
      lcd_draw_box(0,0,83,8,PIXEL_XOR);

      lcd_write_string(0,2,"Current [nA]");
      sprintf(line1, "%4d", result[0]);
      lcd_write_string(4,4,line1);
      lcd_draw_box(15,28,67,44,PIXEL_OFF);
      lcd_update();

      //record starting time
      time_record now = current_time;
      num_file++;

      //compose header
      file_header metadata = {(unsigned int)MARK1, num_files,
potential,
                  interval, duration, (now.year+YEAR1), now.month,
now.day,
                  now.hour, now.min, now.sec, (unsigned int) MARK2};
      data_log_header(metadata);
```

```
    usart0_send_string("\n\r#running\n\r");

    //obtain DAC code
    DAC_start(potential);
    start_interval_timer(interval,duration);
    LPM0;
}

//record: take a measurement, diaplay and record data
void record(void){
    nsigned int i;
    unsigned char line1[14] = "";

    //read ADC, results stored in global variable result
    ADC_read();

    output = result_1Hz-offset;

    //convert ADC output to current value
    if(range = 0x03)  //Rm = 5M, current in pA
        //result = output/26.453;
        result = output*820/31;
    else if(range = 0x01)   //Rm = 50k, current in nA
        //result = output/264.79;
        result = output*5031/19;
    else if(range = 0x02)   //Rm = 500, current in nA
```

```
            //result = output/2.66;
            result = output*8/3;
      else if(range = 0x00)   //Rm = 5, current in uA
            //result = output/43.85;
            result = output*570/13;

      data_log_value(result, OFF);
      data_log_value(result, ON);

      sprintf(buffer, "%d %u %d %d\n\r",dticks,result_1Hz,output,
result);
      usart0_send_string(buffer);

      if(result_1Hz > result_max)
            result_max = result_1Hz;
}

//pause: stops a recording, back to ready mode
void pause(void){
      unsigned int timeout = 5;           //5 minutes timeout
duration
      unsigned char line[10] = "";

      end_measure();
      delay_ms(10);
```

```
        usart0_send_string("\n\r#pause\n\r");
        mode = READY;

        //display result on LCD
        sprintf(line, "max:%d, result_max");
        lcd_write_small_string(1,1,line);
        lcd_draw_line(0,8,83,8,PIXEL_ON);
        lcd_update();

        start_interval_timer(interval, timeout);
}

//done: finish a recording, back to ready mode and display result
void done(void){
        unsigned int timeout = 5;               //5 minutes timeout
duration

        end_measure();
        delay_ms(10);
        usart0_send_string("\n\r#Done\n\r");

        //display result on LCD
        sprintf(line, "max:%d, result_max");
        lcd_write_small_string(1,1,line);
        lcd_draw_line(0,8,83,8,PIXEL_ON);
        lcd_update();
        mode = READY;

        start_interval_timer(interval, timeout);
}

//top of setup menu
void setup(void){
        lcd_clear();

        display8x8(1,1,up);
        display24x24(0,2,setup24);
        lcd_write_string(5,3,"Setup");
        display8x8(1,5,down);
        lcd_update();
}

//top of logs menu
void logs(void){
        lcd_clear();
        display8x8(1,1,up);
        diaplay24x24(0,2,file24);
        lcd_write_string(5,3,"Logs");
        display8x8(1,5,down);
        lcd_update();
```

```
	file_current = data_pointer;
}


//top of clock menu
void clock(void){
	lcd_clear();
	display8x8(1,1,up);
	diaplay24x24(0,2,clock4);
	lcd_write_string(5,3,"Time");
	display8x8(1,5,down);
	lcd_update();
}


void set_interval(void){
	unsigned char line[14] = "";

	lcd_clear();
	lcd_write_string(2,0,"DCPA setup");
	lcd_draw_box(0,0,83,8,PIXEL_XOR);
	lcd_write_string(0,2,"Inteval[sec]");

	lcd_clear_line(4);
	sprintf(line, "%2d", interval);
	lcd_write_string(4,4,line);
	lcd_draw_box(15,28,67,44, PIXEL_OFF);
	lcd_update();
```

```
}

void inc_interval(void){
      unsigned char line[14]="";

      if(++interval ==60){interval = 1;}

      lcd_clear_line(4);
      sprintf(line, "%2d", interval);
      lcd_write_string(4,4,line);
      lcd_update();
}

void dec_interval(void){
      unsigned char line[14] = "";

      if(--interval==0){interval = 59;}
      lcd_clear_line(4);

      sprintf(line, "%2d", interval);
      lcd_write_string(4,4,line);
      lcd_update();
}

void set_duration(void){
      unsigned char line[14] = "";
```

```
	lcd_write_string(0,2,"Duration[min]");

	sprintf(line, "%2d", duration);
	lcd_clear_line(4);
	lcd_write_string(4,4,line);
	lcd_draw_box(15,28,67,44,PIXEL_OFF);
	lcd_update();
}

void inc_duration(void){
	unsigned char line[14] = "";

	if(++duration == 61){duration = 1;}

	lcd_clear_line(4);
	sprintf(line, "%2d", duration);
	lcd_write_string(4,4,line);
	lcd_update();
}

void dec_duration(void){
	unsigned char line[14] = "";

	if(--duration == 0){duration = 60;}

	lcd_clear_line(4);
```

```
      sprintf(line, "%2d", duration);
      lcd_write_string(4,4,line);
      lcd_update();
}

void set_potential(void){
      unsigned char line[14] = "";

      lcd_write_string(0,2, "Potential[mV]");
      sprintf(line, "%4d", potential);
      lcd_clear_line(4);
      lcd_write_string(4,4,line);
      lcd_draw_box(15,28,67,44, PIXEL_OFF);

      lcd_update();
}

void inc_potential(void){
      unsigned char line[14] = "";

      if(++potential > 1200)
            potential = 1200;

      sprintf(line, "%4d", potential);
      lcd_clear_line(4);
      lcd_write_string(4,4,line);
```

```
        lcd_update();
}

void dec_potential(void){
        unsigned char line[14] = "";

        if(--potential == -1200)
                potential = -1200;

        sprintf(line, "%4d", potential);
        lcd_clear_line(4);
        lcd_write_string(4,4,line);
        lcd_update();
}

void set_range(void){
        lcd_clear_line(2);
        lcd_clear_box(15,28,67,44);

        if(range = 0x03){       //Rm = 5M, current in pA
                lcd_write_string(0,2,"Range[pA]");
                lcd_write_string(3,4,"5-1000");
        }
        else if(range = 0x01){  //Rm = 50k, current in nA
                lcd_write_string(0,2,"Range[nA]");
                lcd_write_string(3,4,"0.1-100");
```

```
      }
      else if(range = 0x02){  //Rm = 500, current in nA
            lcd_write_string(0,2,"Range[nA]");
            lcd_write_string(3,4,"10-10000");
      }
      else if(range = 0x00)   //Rm = 5, current in uA
            lcd_write_string(0,2,"Range[uA]");
            lcd_write_string(3,4,"1-1000");
      }

      lcd_draw_box(10,28,72,44, PIXEL_OFF);
      lcd_update();
}

void inc_range(void){

      if(++range == 0x03)
            range = 0x03;

      if(range = 0x03){       //Rm = 5M, current in pA
            lcd_write_string(0,2,"Range[pA]");
            lcd_write_string(3,4,"5-1000");
      }
      else if(range = 0x01){  //Rm = 50k, current in nA
            lcd_write_string(0,2,"Range[nA]");
            lcd_write_string(3,4,"0.1-100");
```

```
        }
        else if(range = 0x02){  //Rm = 500, current in nA
                lcd_write_string(0,2,"Range[nA]");
                lcd_write_string(3,4,"10-10000");
        }
        else if(range = 0x00)   //Rm = 5, current in uA
                lcd_write_string(0,2,"Range[uA]");
                lcd_write_string(3,4,"1-1000");
        }

        lcd_draw_box(10,28,72,44, PIXEL_OFF);
        lcd_update();
}

void dec_range(void){

        if(--range == 0x00)
                range = 0x00;

        if(range = 0x03){       //Rm = 5M, current in pA
                lcd_write_string(0,2,"Range[pA]");
                lcd_write_string(3,4,"5-1000");
        }
        else if(range = 0x01){  //Rm = 50k, current in nA
                lcd_write_string(0,2,"Range[nA]");
                lcd_write_string(3,4,"0.1-100");
```

```
      }
      else if(range = 0x02){  //Rm = 500, current in nA
            lcd_write_string(0,2,"Range[nA]");
            lcd_write_string(3,4,"10-10000");
      }
      else if(range = 0x00)   //Rm = 5, current in uA
            lcd_write_string(0,2,"Range[uA]");
            lcd_write_string(3,4,"1-1000");
      }

      lcd_draw_box(10,28,72,44, PIXEL_OFF);
      lcd_update();
}

void delay_ms(unsigned int delay){
    while (delay--){
      __delay_cycles(32);
    }
}

//using the watchdog timer for interval timing of samples
//sample intervals = 1 to 60 sec, duration in minutes
//watchdog is clocked by ACLK
void start_interval_timer(int ival, int dur){
      nticks = 0;
      dticks = 0;
      dsec = dmin = 0;

      ntimes = ival;
      dtimes = dur*60;

      WDTCTL = WDTPW + WDTHOLD;                 //stop watch dog
timer
      IE1 &= ~WDTIE;                                  //diable WDT
interrupt
      //WDT_ADLY_1000==(WDTPW+WDTTMSEL+WDTCNTCL+WDTSSEL)
      WDTCTL = WDT_ADLY_1000;                   //WDT 1S, ACLK,
INTERVAL MODE
      //WDTISx bit: where to change the sample rate
      IE1 |= WDTIE;                                   //enable WDT
interrupt
}

void stop_interval_timer(void){
      WDTCTL = WDTPW + WDTHOLD;                 //stop watchdog timer
      IE1 &= ~WDTIE;                                  //diable WDT
interrupt
}

#pragma vector=WDT_VECTOR
__interrupt void interval_timer_ISR (void){
    if(++nticks == ntimes){

        event_tray |= FIVAL;

        nticks = 0;

    }


    if(++dticks > dtimes){

        event_tray |= FDONE;

        dticks = 0;

    }


    if(++dsec == 60){

        dsec = 0;

        dmin++;

    }


    LPM3_EXIT;


}


//background process: respond to commands received from usart0

void respond_to_cmd(void){


    //echo command

    usart0_send_char(usart0_rx_buffer[0]);

    usart0_send_char(usart0_rx_buffer[1]);

    usart0_send_char(usart0_rx_buffer[2]);
```

```
        //display help message

        if(receive_command('h','p')){usart0_send_string(helpMsg);}

        //display current setting

        else if(receive_command('g','s'){get_status();}

        //update potentiostat parameters

        else if(receive_command('s','p'){set_params();}

        //get time

        else if(receive_command('g','t'){get_time();}

        //set time

        else if(receive_command('s','t'){set_time();}

        //get file headers

        else if(receive_command('g','h'){data_log_summary();}

        //display file

        else if(receive_command('g','f'){data_log_read();}

        //delete stored files

        else if(receive_command('s','c'){data_log_erase();

get_status();}

        //start a miniEC experiment

        else if(receive_command('m','r')){

                //put B2 events in the event tray

                //monitor_flag = ON;

                event_tray |= FB3;

        }

        //stop a miniEC experiment

        else if(receive_command('m','s')){
                //put B1 events in the event tray
                event_tray |= FB1; //monitor_flag = OFF;
        }
        //default:display help message
        else{usart0_send_string("\n\r#Command not valid. Type hp for
menu\n\r");}

        //reset buffers and enable rx interrupts
        memset(usart0_rx_buffer, 0, RX_MSG_MAX);
        memset(usart0_tx_buffer, 0, TX_MSG_MAX);
        usart0_rx_flag = OFF;
        usart0_rx_index = 0;
        usart0_rx_isr_enable();

}

//analog and digital conversoins
//DAC initialize and calibrate
void DAC_init(void){

        SD16CTL |= SD16REFON + SD16VMIDON;                     //
Internal 1.2V ref on, availble for external use.
        DAC12_1CTL = DAC12IR + DAC12SREF_3 + DAC12AMP_5 +  // SD16 Ref,
                       DAC12OPS + DAC12ENC;
      DAC12_1CTL |=  DAC12CALON;                          // calibration
DAC1
```

```
    delay_ms(32);                                     //wait for
calibration

}

//write a value for DAC to output
//0xFFF = 1200mV
void DAC_start(int potential){
      dacValue = 2400/0xFFF*(potential+1200);
      DAC12_1DAT = dacValue;                          //need to check
}

//switch off DAC12
void DAC_stop(void){
      DAC12_1CTL &= ~DAC12ENC;                  //disable DAC1
conversion
      DAC12_1CTL = 0;                                 //switch off
DAC12
}

void ADC_init(void){

      SD16CTL |= SD16VMIDON+SD16REFON+SD16SSEL0; // 1.2V ref, MCLK
      SD16INCTL0 |= SD16INTDLY_0;               // Interrupt on 4th
sample
```

```
      SD16CCTL0 |= SD16BUF_1+SD16IE ;           // Median speed
buffer, Enable interrupt
      SD16CCTL0 |= SD16OSR_512;
    SD16INCTL0 |= SD16GAIN0+SD16GAIN2;          //preamplifier = * 32
      delay_ms(32);                                         //wait
for calibration

      //get internal offset
      SD16INCTL0 = SD16INCH_7;
      SD16CCTL0 |= SD16SC;
      //LPM0;           //CPUOFF
      while(!(SD16CCTL0 & SD16IFG));
      initial = SD16MEM0;

}

//ADC read
//for adc measurements using oversampling and averaging methods for
noise
//suppression. ADC value then stored/displayed/transmitted via usart
void ADC_read(void){
      SD16INCTL0 = SD16INCH_4;
      SD16CCTL0 |= SD16SC;                      // Set bit to start
conversion

      __bis_SR_register(LPM0_bits+GIE);        // Enter LPM0
```

```
}

void ADC_stop(void){

      while(!(SD16CCTL0 & SD16IFG));      //wait for conversion to
stop
      SD16CCTL0 &= ~SD16SC;               //disable ADC
      SD16CTL &= ~(SD16VMIDON+SD16REFON+SD16SSEL0);
      //switch off ADC & ref voltage
      //SD16IFG = 0;                            //clear any pending
interrupts
}

#pragma vector=SD16A_VECTOR
__interrupt void SD16ISR(void)
{
  switch (SD16IV)
  {
  case 2:                                   // SD16MEM Overflow
    break;
  case 4:                                   // SD16MEM0 IFG
    result_1Hz = SD16MEM0;   // Save CH0 results (clears IFG)

  }
  LPM0_EXIT;
```

```
}


void end_measure(void){
      stop_interval_timer();
      ADC_stop();
      DAC_stop();


}


//usart0
void usart0_init(void){


      UCA0CTL1 |= UCSSEL_1;                     // CLK = ACLK
      UCA0BR0 = 0x03;                           // 32k/9600 - 3.41
      UCA0BR1 = 0x00;                           //
      UCA0MCTL = 0x06;                          // Modulation
      UCA0CTL1 &= ~UCSWRST;                     // **Initialize USCI
state machine**
      IE2 |= UCA0RXIE;                          // Enable USCI_A0 RX
interrupt


      _BIS_SR(LPM3_bits + GIE);                 // Enter LPM3,
interrupts enabled
}


//transmit 1 byte
```

```
void usart0_send_char(char c){
      while(!(IFG2&UCA0TXIFG));                        //USART0 TX
buffer ready
      UCA0TXBUF = c;
}

//transmit messages through usart0
void usart0_send_string(char msg[]){
      while(*msg!= 0){
            while((IFG2&UCA0TXIFG) == 0);       //USART TX buffer
ready?
            UCA0TXBUF = *msg;                               //send
character to TXBUF
            msg++;
      //increment pointer
      }
}

//send numbers
void usart0_send_int(int value){
      char buffer[32];

      if(value == 0){
            usart0_send_char('0');
            usart0_send_char(' ');
      }
```

```
    else{
        sprintf(buffer, "%d", value);
        usart0_send_string(buffer);
    }
}

void usart0_rx_isr_enable(void){
    IE2 |= UCA0RXIE;                          // Enable USCI_A0 RX
interrupt
}

void usart0_rx_isr_disable(void){
    IE2 &= ~UCA0RXIE;        // disable USCI_A0 RX interrupt
}

//  Echo back RXed character, confirm TX buffer is ready first
#pragma vector=USCIAB0RX_VECTOR
__interrupt void usart0_rx_isr (void)
{
    volatile unsigned char temp;

    //check status register for receive errors
    if(UCA0STAT & UCRXERR){
        //clear error flags by forcing a dummy read
        temp = UCA0RXBUF;
        return;
```

```
      }

      while(!(IFG2 & UCA0TXIFG));
      usart0_rx_buffer[usart0_rx_index] = UCA0RXBUF;
      //UCA0RXBUF to array
      usart0_rx_index++;

      //UCA0TXBUF = UCA0RXBUF;                    // TX -> RXed
character
      //check for carriage return character or end of buffer
      if((usart0_rx_index == RX_MSG_MAX) ||
         (usart0_rx_buffer[usart0_rx_index-1] == CR)){
            usart0_rx_isr_disable();        // disable USCI_A0 RX
interrupt
            usart0_rx_flag = ON;    //turn RX flag on??
      }
      LPM3_EXIT;
}

//flash memory
void flash_write(void *dst, const void*src, unsigned int size){
      FCTL3 = FWKEY;                  //UNLOCK
      FCTL1 = FWKEY|WRT;              //select write

      while(size){
            size--;
```

```
		((unsigned char *)dst)[size] = ((unsigned char
*)src)[size];
		while(FCTL3&BUSY){}		//wait until write is
completed
	}
	FCTL1 = FWKEY;
	FCTL3 = FWKEY|LOCK;
	return;
}

void flash_write_word(void *dst, const void *src, unsigned int size){
	FCTL3 = FWKEY;
	FCTL1 = FWKEY|WRT;

	//calculate size in words: divide by 2
	size >>= 1;
	while(size){
		size--;
		((unsigned short *)dst)[size] = ((unsigned short
*)src)[size];
		while(FCTL3&BUSY){}
	}
	FCTL1 = FWKEY;
	FCTL3 = FWKEY|LOCK;
	return;
}
```

```
void flash_erase_segment(void *address){
      FCTL3 = FWKEY;
      FCTL1 = FWKEY|ERASE;
      *(unsigned short*)address = 0xffff;
      FCTL1 = FWKEY;
      FCTL3 = FWKEY|LOCK;
      return;
}

void flash_clar_memory(void){
      unsigned int *address = (unsigned int *)DATA_START;

      dint();
      do{
            if((unsigned int)address & 0x1000)
                  LED2 |= ON;
            else  LED2 = OFF;

            FCTL1 = FWKEY + ERASE;
            *address = 0x00;
            address += 0x0100;
      }while(address<(unsigned int *)DATA_END);

      FCTL3 = FWKEY|LOCK;
      eint();
      return;
}

void write_SegA(char value){
      char *Flash_ptr;
      unsigned int i;

      Flash_ptr = (char *)0x1080;         //initialize flash pointer
      FCTL1 = FWKEY + ERASE;
      FCTL3 = FWKEY;
      *Flash_ptr = 0;                           //dummy write to
erase flash segment

      FCTL1 = FWKEY + WRT;                //set WRT bit for write
operation

      for(i=0; i<128; i++)
            *Flash_ptr++ = value;         //write value to flash

      FCTL1 = FWKEY;
      FCTL3 = FWKEY|LOCK;
      return;
}

//assemble data file header and write it to file
void logHeader(int mark1, int mark2){
      int filemark = mark1;
      int headmark = mark2;

      flash_write_word(data_pointer, filemark);       //write result
to file
      flash_write_word(++data_pointer, startTime.secs);
      flash_write_word(++data_pointer, startTime.mins);
      flash_write_word(++data_pointer, startTime.hours);
      flash_write_word(++data_pointer, startTime.days);
      flash_write_word(++data_pointer, startTime.month);
      flash_write_word(++data_pointer, startTime.year);
      flash_write_word(++data_pointer, potential);
      flash_write_word(++data_pointer, interval);
      flash_write_word(++data_pointer, duration);
      flash_write_word(++data_pointer, range);
      flash_write_word(++data_pointer, headmark);
      datapointer = ++data_pointer;
      mem_left = mem_left - 22;
}

void logData(int mydata){
      if(datapointer < (unsigned int *)DATA_END){
            flash_write_word(data_pointer, mydata);
            data_pointer++;
            mem_left = mem_left - 2;
      }
```

```
}


void logTail(int marker){

      if(data_pointer < (unsigned int *)DATA_END){

            flash_write_word(data_pointer, marker);

            flash_write_word(++data_pointer, secs);

            flash_write_word(++data_pointer, mins);

            flash_write_word(++data_pointer, hours);

            flash_write_word(++data_pointer, days);

            flash_write_word(++data_pointer, month);

            flash_write_word(++data_pointer, year);

            data_pointer++;

            mem_left = mem_left - 14;

      }

}


int * find_next_location(void){

      unsigned int *nextptr = (unsigned int *)DATA_START;

      unsigned int *lastptr = (unsigned int *)DATA_END;

      unsigned int counter = 0;

      numFiles = 0;

      totalBytes = (unsigned int)lastptr - (unsigned int)nextptr;


      while(nextptr <(unsigned int *)DATA_END){

            if(*nextptr == 1111)

                  numFiles++;
```

```
        if((*nextptr - 0xFFFF) == 0){
            bytesLeft = bytesLeft - counter * 2;
            break;
        }
        nextptr++;
        counter++;
    }
    if(nextptr == lastptr){
        counter++;
        mem_left = mem_left - counter * 2;
    }
    return nextptr;
}

//LCD functions
void lcd_send(unsigned char data, unsigned char cd){

    //check for command or data
    if(cd==SEND_CHR)
        P3OUT |=  SPI_MISO;
    else
        P3OUT &= ~SPI_MISO;

    //enable display controller (active low)
    P3OUT &= ~LCD_CS;
```

```
      //send via SPI
      UCB0TXBUF = data;                         //send data

      //while((UCB0STAT & UCBUSY) == 0);  //wait for ready U0TXBUF
      while(!(IFG2 & UCB0TXIFG));

      P3OUT |= LCD_CS;                          //disable display
controller

}
void lcd_clear(void){
      int i;
      //loop all cache array
      for (i=0; i<LCD_CACHE_SIZE; i++)
            LcdMemory[i] = 0;
}

//update the lcd screen after modification
void lcd_update(void){
      int i;

      //send base address to X=0 Y=0
      lcd_send(0x80, SEND_CMD);
      lcd_send(0x40, SEND_CMD);

      //send contents of memory buffer to LCD
      for(i=0; i<LCD_CACHE_SIZE; i++)
            lcd_send(LcdMemory[i], SEND_CHR);
}

//MCU SPI and LCD controller initialization
void lcd_init(void){
      //pull-up on LCD reset pin
      P3OUT |= LCD_RES;
      //P5OUT |= LCD_RES;

      //Setup MCU ports for SPi connection
      P3DIR |= 0x0F;                      //P3.0,1,2,3 output
direction
      P3DIR |= 0x10;                      //P3.4 RES



      //toggle LCD reset pin
      P3OUT &= ~LCD_RES;
      __delay_cycles(32000);
      P3OUT |= LCD_RES;

      //init SPI
      UCB0CTL1 = UCSWRST;
      //Mode: master, MSB first, Sync spi, clk phase
      UCB0CTL0 |= UCMST + UCMSB + UCSYNC + UCCKPH;
      UCB0CTL0 &= ~UC7BIT;    //8bits
      UCB0CTL0 &= ~UCCKPL;    //inactive state: low
      UCB0CTL1 |= UCSSEL_1;   //ACLK
      UCB0BR0 = 0x04;               //Baud rate = ACLK/4
      UCB0BR1 = 0;
      UCB0CTL1 &= ~UCSWRST;   //enable SPI

      P3SEL |= BIT1 + BIT3;         //P3.1,3 SPI MOSI, CLK
      //disable LCD controller
      P3OUT |= LCD_CS;
      __delay_cycles(32000);

      //send sequence of commands
      lcd_send(0x21 , SEND_CMD);          //switch to LCD extended
commands
      lcd_send(0xC8 , SEND_CMD);          //set LCD Vop(Contrast)
      lcd_send(0x06 , SEND_CMD);          //Set Temp coefficient
      lcd_send(0x13 , SEND_CMD);          //LCD biase mode 1:48
      lcd_send(0x20 , SEND_CMD);          //switch back to LCD
Standard commands
      //lcd_send(0x08 , SEND_CMD);        //LCD blank
      lcd_send(0x09, SEND_CMD);           //LCD all segments all
      delay_ms(100);
      lcd_send(0x0C , SEND_CMD);          //LCD in normal mode

      //clear and update
      lcd_clear();
```

```
    lcd_update();

}

//x range 0-83, y range 0-47
//display a character at position x,y on the LCD with basic 5*7 font
used
void lcd_write_char(unsigned char x, unsigned char y, unsigned char
ch){
    unsigned int idx = 0;
    unsigned int i = 0;

    //check for out of range positions
    if (x > LCD_X_RES) return;
    if (y > LCD_Y_RES) return;

    idx = (x*48 + y*48*14)/8;

    for (i=0; i<5; i++){
        LcdMemory[idx] = FontLookup[ch-32][i]<<1;            //??
        idx++;
    }
}


//display a string on starting at position col, row
```

```
//basic 5*7 font
void lcd_write_string(unsigned char col, unsigned char row, unsigned
char *dataPtr){
	//loop to the end of string
	while (*dataPtr){
		lcd_write_char(col, row, (*dataPtr));
		col++;
		dataPtr++;
	}
}

//set, clear or invert a pixel at location x,y
void lcd_set_pixel(unsigned char x, unsigned char y, unsigned char
mode){
	unsigned int idx = 0;
	unsigned char offset = 0;
	unsigned char data = 0;

	//check for out of range
	if (x > LCD_X_RES) return;
	if (y > LCD_Y_RES) return;

	idx = ((y/8)*84)+x;
	offset = y-((y/8)*8);

	data = LcdMemory[idx];
```

```
      if (mode == PIXEL_OFF)
            data &= (~(0x01<<offset));
      else if (mode == PIXEL_ON)
            data |= (0x01 << offset);
      else if ( mode == PIXEL_XOR)
            data ^= (0x01<<offset);

      LcdMemory[idx] = data;
}
//set cursor location to xy
void lcd_set_xy(unsigned char x, unsigned char y){
      LcdMemIdx = x*6 + y*84;
}

//set LCD contrast
void lcd_set_contrast(unsigned char contrast){
      //LCD extended commands
      lcd_send(0x21, SEND_CMD);

      //set LCD Vop (Contrast)
      lcd_send(0x80|contrast, SEND_CMD);

      //LCD standard commands, horizontal addressing mode
      lcd_send(0x20, SEND_CMD);
}
```

```
//draw bitmap image on LCD using the whole area
void lcd_fill_image(const unsigned char * bitmap){
      static unsigned int i;

      lcd_clear();
      lcd_update();

      for (i=0; i<LCD_CACHE_SIZE; i++)
            lcd_send(bitmap[i], SEND_CHR);

      lcd_set_xy(0,0);
      __delay_cycles(32000);
}

//draws a line between two points on the display
//borrowed from Sylvain Bissonnette's page at http://www.microsyl.com
void LcdLine(unsigned char x1, unsigned char y1, unsigned char x2,
unsigned char y2, unsigned char mode){
      int dx, dy, stepx, stepy, fraction;

      dy = y2-y1;
      dx = x2-x1;

      if(dy < 0) {dy = -dy; stepy = -1;}
      else stepy = 1;
```

```
if(dx < 0) {dx = -dx; stepx = -1;}
else stepx = 1;

dx <<= 1;
dy <<= 1;

lcd_set_pixel(x1, y1, mode);

if(dx > dy){
        fraction = dy - (dx>>1);
        while (x1 != x2){
                if (fraction >= 0){
                        y1 += stepy;
                        fraction -= dx;
                }
                x1 += stepx;
                fraction += dy;
                lcd_set_pixel(x1, y1, mode);
        }
}
else{
        fraction = dx - (dy>>1);
        while (y1 != y2){
                if (fraction >= 0){
                        x1 += stepx;
                fraction -=dy;
            }
            y1 += stepy;
            fraction += dx;
            lcd_set_pixel(x1, y1, mode);
        }
    }
}

//draw a box outline, filled box or invert pixels in box area
void lcd_draw_box(unsigned char x1, unsigned char y1, unsigned char
x2, unsigned char y2, unsigned char mode){
    unsigned char i,j;

    switch(mode){
        case(BOX_OFF):          //box outline
            for (j=x1; j<=x2; j++){
                lcd_set_pixel(j,y1,PIXEL_ON);
                lcd_set_pixel(j,y2,PIXEL_ON);
            }
            for (i=y1; i<=y2; i++){
                lcd_set_pixel(x1,i,PIXEL_ON);
                lcd_set_pixel(x2,i,PIXEL_ON);
            }
            break;
            case(BOX_ON):           //filled box
                  for (i=y1; i<=y2; i++){
                        for (j=x1; j<=x2; j++)
                              lcd_set_pixel(j,i,PIXEL_ON);
                  }
                  break;

            case(BOX_XOR):          //inverted pixels box
                  for (i=y1; i<=y2; i++){
                        for (j=x1; j<=x2; j++)
                              lcd_set_pixel(j,i,PIXEL_XOR);
                  }
                  break;

            default: break;
      }
}

//set pixels defined by box to background
void lcd_clear_box(unsigned char x1, unsigned char y1, unsigned char
x2, unsigned char y2){
      unsigned char i, j;

      for (i=y1; i<=y2; i++){
            for (j=x1; j<=x2; j++)
                  lcd_set_pixel(j,i,PIXEL_OFF);
      }
}

//LcdLine wrapped to follow this file's naming conventions
void lcd_draw_line(unsigned char x1, unsigned char y1, unsigned char
x2, unsigned char y2, unsigned char mode){
      LcdLine(x1,y1,x2,y2,mode);
}


//get a current time and date as a string
//format as hh:mm:ss, YYYY-MM-DD
void rtc_time_string(unsigned char *buffer){
      unsigned char *time_buffer;
      time_record now;

      time_buffer = buffer;
      now = current_time                  //get current time

      //serialize date and time
      time_buffer += sprintf(time_buffer, "%04d-%02d-
%02d %02d:%02d:%02d",
                  now.year+YEAR1,   now.month, now.day, now.hour,
now.min, now.sec);
}
```

```
#pragma vector=BASICTIMER_VECTOR
__interrupt void basic_timer_isr (void)
{
      if(++currentTime.secs == 60){
            currentTime.secs = 0;
            if(++currentTime.mins == 60){
                  currentTime.mins = 0;
                  if(++currentTime.hours ==24){
                        currentTime.hours = 0;
                        if(++currentTime.days == 32){       //count
days up to 1 month
                              currentTime.days = 1;
                        }
                        else if(currentTime.days == 31){    //for 30
day month
                              if((currentTime.month ==
4)||(currentTime.month == 6)||
                              (currentTime.month ==
9)||(currentTime.month == 11)){
                                    currentTime.days = 1;
                                    currentTime.month++;
                              }
                        }
                        //for month = feb and leap year
                        else if((currentTime.month ==
2)&&(currentTime.days == 30)){
```

```
                    currentTime.days = 1;
                    currentTime.month++;
                }
                //for month = Feb, not a leap year
                else if((currentTime.month ==
2)&&(currentTime.days == 29)&&
                !(LEAP(currentTime.year))){
                    currentTime.days = 1;
                    currentTime.month++;
                }
                if(++currentTIme.month == 13){
                    currentTime.month = 1;
                    currentTime.year++;
                }
            }
        }
    }
    LPM3_EXIT;
}

void InitializeSystem(void){
    //system clock setup
    /* - set the system clock to run at 8MHz in active mode
     * - an external crystal sets the auxiliary clock ACLK to
32768Hz
```

```
       * - the master (MCLK) and sub-main clocks (SMCLK) are set by
the
       *   internally controlled digital clock (DCOCLK) to 8MHz
       * - MCLK = SMCLK = DCO = (n+1) x ACLK
      */
      WDTCTL = WDTPW + WDTHOLD;           //stop watchdog timer
      SCFI0 |= FN_4;                            //x2 DCO frequency,
8MHz nominal DCO
      SCFQCTL = 121;                            //(121+1) x 32768 x 2
= 7.995 MHz
      FLL_CTL0 = DCOPLUS + XCAP18PF;      // DCO+ set so freq = xtal
x D x N+1
      delay_ms(25);
    FLL_CTL0 |= XCAP18PF;

      //initialize basic timer to real time clock
      IE2 |= BTIE;                              // enable basic timer
interrupt
      BTCTL = BT_ADLY_1000;               // set timer to 1 second
intervals

      //all unused pins, switched to port function, OUTPUT direction
      //port1 setup: P1.2,3  SD16 channel 4
      //             P1.4 DAC1
      //             P1.5 Analog Switch
      P1SEL |= BIT2+BIT3+ BIT4;                // enable SD16 channel
4, A/D channel 1
      P1DIR |= 0x00;
      P1DIR &= ~BIT2;
      P1DIR &= ~BIT3;
      P1OUT = 0x00;
      P1OUT |= BIT5;                                  // P1.5 high,
switch off

      //port2 setup: P2.4, 2.5 = USART0 TXD/RXD
      //             P2.0, 2.1, 2.2, 2.3 Switches
      P2IFG = 0x00;                             //clear pending
interrupts on port1
      P2IES = B1 + B2 + B3 + B4;          //interrupt on falling edge
      P2IE = B1 + B2 + B3 + B4;           //enable interrupts

      P2SEL |= BIT4 + BIT5;
      P2DIR |= BIT4;
      P2DIR &= ~BIT5;                           // BIT5 input
direction

      //port3 setup: LCD
      P3SEL |= BIT1 + BIT3;
      P3DIR |= BIT4 + BIT0;

      //port4 setup: P4.0&P4.1 LED1&LED2
```

```
        P4DIR |= LED1 + LED2;

        P4OUT |= LED1 + LED2;               //LED 1&2 OFF


        //port6 setup:

        //            P6.0, P6.1 SMUX control

        P6DIR |= BIT0 + BIT1;

        P6OUT = 0x00;


        //initialize LCD

        lcd_init();

        lcd_set_contrast(0x45);

        delay_ms(1);*/


        //initialize USART

        usart0_init();


        lcd_clear();

        lcd_update();

        delay_ms(1);

   }
```

A.2 LCD driver interface

```
#ifndef PCD8544_H_

#define PCD8544_H_
```

```
#define ON        1
#define OFF       0
#define TRUE      1
#define FALSE     0

#define SEND_CMD  0
#define SEND_CHR  1

#define LCD_X_RES 84
#define LCD_Y_RES 48

#define PIXEL_OFF 0
#define PIXEL_ON  1
#define PIXEL_XOR 2

#define BOX_OFF         0
#define BOX_ON          1
#define BOX_XOR         2

#define FONT_0X         0
#define FONT_1X         1
#define FONT_2X         2

#define LCD_CACHE_SIZE  ((LCD_X_RES * LCD_Y_RES) / 8)

//SPI
```

```
#define LCD_RES         BIT4
#define LCD_CS          BIT0
#define SPI_MISO  BIT2
#define SPI_MOSI  BIT1
#define SPI_SCKL  BIT3

//VARIABLES
unsigned int LcdMemIdx;                               //LCD memory index
unsigned char LcdMemory[LCD_CACHE_SIZE];  //LCD memory

//Function Prototypes
void lcd_init(void);
void lcd_clear(void);
void lcd_update(void);
void lcd_clear_line(unsigned char row);
void lcd_send(unsigned char data, unsigned char cd);
void lcd_write_char(unsigned char x, unsigned char y, unsigned char ch);
void lcd_write_small_char(unsigned char x, unsigned char y, unsigned char ch);
void lcd_write_string(unsigned char col, unsigned char row, unsigned char *dataPtr);
void lcd_write_small_string(unsigned char col, unsigned char row, unsigned char *dataPtr);
void lcd_write_2X_char(unsigned char x, unsigned char y, unsigned
char ch);
void lcd_write_2X_string(unsigned char col, unsigned char row,
unsigned char *dataPtr);
void lcd_set_pixel(unsigned char x, unsigned char y, unsigned char
mode);
void lcd_set_xy(unsigned char x, unsigned char y);
void lcd_set_contrast(unsigned char contrast);
void lcd_fill_image(const unsigned char * bitmap);
void LcdLine(unsigned char x1, unsigned char y1, unsigned char x2,
unsigned char y2, unsigned char mode);
void lcd_draw_box(unsigned char x1, unsigned char y1, unsigned char
x2, unsigned char y2, unsigned char mode);
void lcd_clear_box(unsigned char x1, unsigned char y1, unsigned char
x2, unsigned char y2);
void lcd_draw_line(unsigned char x1, unsigned char y1, unsigned char
x2, unsigned char y2, unsigned char mode);

//Standard ASCII Characters in a 5*7 pixel font
//Characters 32-125 in ASCII table
static const unsigned char FontLookup[][5]={

      { 0x00 , 0x00 , 0x00 , 0x00 , 0x00 } , // sp
      { 0x00 , 0x00 , 0x2f , 0x00 , 0x00 } , // !
      { 0x00 , 0x07 , 0x00 , 0x07 , 0x00 } , // ”
      { 0x14 , 0x7f , 0x14 , 0x7f , 0x14 } , // #
      { 0x24 , 0x2a , 0x7f , 0x2a , 0x12 } , // $
      { 0xc4 , 0xc8 , 0x10 , 0x26 , 0x46 } , // %
      { 0x36 , 0x49 , 0x55 , 0x22 , 0x50 } , // &
      { 0x00 , 0x05 , 0x03 , 0x00 , 0x00 } , // ’
      { 0x00 , 0x1c , 0x22 , 0x41 , 0x00 } , // (
      { 0x00 , 0x41 , 0x22 , 0x1c , 0x00 } , // )
      { 0x14 , 0x08 , 0x3E , 0x08 , 0x14 } , // _
      { 0x08 , 0x08 , 0x3E , 0x08 , 0x08 } , // +
      { 0x00 , 0x00 , 0x50 , 0x30 , 0x00 } , // ,
      { 0x10 , 0x10 , 0x10 , 0x10 , 0x10 } , // -
      { 0x00 , 0x60 , 0x60 , 0x00 , 0x00 } , // .
      { 0x20 , 0x10 , 0x08 , 0x04 , 0x02 } , // /
      { 0x3E , 0x51 , 0x49 , 0x45 , 0x3E } , // 0
      { 0x00 , 0x42 , 0x7F , 0x40 , 0x00 } , // 1
      { 0x42 , 0x61 , 0x51 , 0x49 , 0x46 } , // 2
      { 0x21 , 0x41 , 0x45 , 0x4B , 0x31 } , // 3
      { 0x18 , 0x14 , 0x12 , 0x7F , 0x10 } , // 4
      { 0x27 , 0x45 , 0x45 , 0x45 , 0x39 } , // 5
      { 0x3C , 0x4A , 0x49 , 0x49 , 0x30 } , // 6
      { 0x01 , 0x71 , 0x09 , 0x05 , 0x03 } , // 7
      { 0x36 , 0x49 , 0x49 , 0x49 , 0x36 } , // 8
      { 0x06 , 0x49 , 0x49 , 0x29 , 0x1E } , // 9
      { 0x00 , 0x36 , 0x36 , 0x00 , 0x00 } , // :
      { 0x00 , 0x56 , 0x36 , 0x00 , 0x00 } , // ;
      { 0x08 , 0x14 , 0x22 , 0x41 , 0x00 } , // <
      { 0x14 , 0x14 , 0x14 , 0x14 , 0x14 } , // =
      { 0x00 , 0x41 , 0x22 , 0x14 , 0x08 } , // >

      { 0x02 , 0x01 , 0x51 , 0x09 , 0x06 } , // ?

      { 0x32 , 0x49 , 0x59 , 0x51 , 0x3E } , // @

      { 0x7E , 0x11 , 0x11 , 0x11 , 0x7E } , // A

      { 0x7F , 0x49 , 0x49 , 0x49 , 0x36 } , // B

      { 0x3E , 0x41 , 0x41 , 0x41 , 0x22 } , // C

      { 0x7F , 0x41 , 0x41 , 0x22 , 0x1C } , // D

      { 0x7F , 0x49 , 0x49 , 0x49 , 0x41 } , // E

      { 0x7F , 0x09 , 0x09 , 0x09 , 0x01 } , // F

      { 0x3E , 0x41 , 0x49 , 0x49 , 0x7A } , // G

      { 0x7F , 0x08 , 0x08 , 0x08 , 0x7F } , // H

      { 0x00 , 0x41 , 0x7F , 0x41 , 0x00 } , // I

      { 0x20 , 0x40 , 0x41 , 0x3F , 0x01 } , // J

      { 0x7F , 0x08 , 0x14 , 0x22 , 0x41 } , // K

      { 0x7F , 0x40 , 0x40 , 0x40 , 0x40 } , // L

      { 0x7F , 0x02 , 0x0C , 0x02 , 0x7F } , // M

      { 0x7F , 0x04 , 0x08 , 0x10 , 0x7F } , // N

      { 0x3E , 0x41 , 0x41 , 0x41 , 0x3E } , // O

      { 0x7F , 0x09 , 0x09 , 0x09 , 0x06 } , // P

      { 0x3E , 0x41 , 0x51 , 0x21 , 0x5E } , // Q

      { 0x7F , 0x09 , 0x19 , 0x29 , 0x46 } , // R

      { 0x46 , 0x49 , 0x49 , 0x49 , 0x31 } , // S

      { 0x01 , 0x01 , 0x7F , 0x01 , 0x01 } , // T

      { 0x3F , 0x40 , 0x40 , 0x40 , 0x3F } , // U

      { 0x1F , 0x20 , 0x40 , 0x20 , 0x1F } , // V

      { 0x3F , 0x40 , 0x38 , 0x40 , 0x3F } , // W
      { 0x63 , 0x14 , 0x08 , 0x14 , 0x63 } , // X

      { 0x07 , 0x08 , 0x70 , 0x08 , 0x07 } , // Y

      { 0x61 , 0x51 , 0x49 , 0x45 , 0x43 } , // Z

      { 0x00 , 0x7F , 0x41 , 0x41 , 0x00 } , // [

      { 0x55 , 0x2A , 0x55 , 0x2A , 0x55 } , // back slash

      { 0x00 , 0x41 , 0x41 , 0x7F , 0x00 } , // ]

      { 0x04 , 0x02 , 0x01 , 0x02 , 0x04 } , // ˆ

      { 0x40 , 0x40 , 0x40 , 0x40 , 0x40 } , // _

      { 0x00 , 0x01 , 0x02 , 0x04 , 0x00 } , // ’

      { 0x20 , 0x54 , 0x54 , 0x54 , 0x78 } , // a

      { 0x7F , 0x48 , 0x44 , 0x44 , 0x38 } , // b

      { 0x38 , 0x44 , 0x44 , 0x44 , 0x20 } , // c

      { 0x38 , 0x44 , 0x44 , 0x48 , 0x7F } , // d

      { 0x38 , 0x54 , 0x54 , 0x54 , 0x18 } , // e

      { 0x08 , 0x7E , 0x09 , 0x01 , 0x02 } , // f

      { 0x0C , 0x52 , 0x52 , 0x52 , 0x3E } , // g

      { 0x7F , 0x08 , 0x04 , 0x04 , 0x78 } , // h

      { 0x00 , 0x44 , 0x7D , 0x40 , 0x00 } , // i

      { 0x20 , 0x40 , 0x44 , 0x3D , 0x00 } , // j

      { 0x7F , 0x10 , 0x28 , 0x44 , 0x00 } , // k

      { 0x00 , 0x41 , 0x7F , 0x40 , 0x00 } , // l

      { 0x7C , 0x04 , 0x18 , 0x04 , 0x78 } , // m

      { 0x7C , 0x08 , 0x04 , 0x04 , 0x78 } , // n

      { 0x38 , 0x44 , 0x44 , 0x44 , 0x38 } , // o

      { 0x7C , 0x14 , 0x14 , 0x14 , 0x08 } , // p

      { 0x08 , 0x14 , 0x14 , 0x18 , 0x7C } , // q
```

```
    { 0x7C , 0x08 , 0x04 , 0x04 , 0x08 } , // r
    { 0x48 , 0x54 , 0x54 , 0x54 , 0x20 } , // s
    { 0x04 , 0x3F , 0x44 , 0x40 , 0x20 } , // t
    { 0x3C , 0x40 , 0x40 , 0x20 , 0x7C } , // u
    { 0x1C , 0x20 , 0x40 , 0x20 , 0x1C } , // v
    { 0x3C , 0x40 , 0x30 , 0x40 , 0x3C } , // w
    { 0x44 , 0x28 , 0x10 , 0x28 , 0x44 } , // x
    { 0x0C , 0x50 , 0x50 , 0x50 , 0x3C } , // y
    { 0x44 , 0x64 , 0x54 , 0x4C , 0x44 } , // z
    { 0x08 , 0x6C , 0x6A , 0x19 , 0x08 } , // {
    { 0x30 , 0x4E , 0x61 , 0x4E , 0x30 } , // |
    { 0x7E , 0x7E , 0x7E , 0x7E , 0x7E }   // }
} ;


// Standard ASCII characters in a 3*5 pixel font
// Numbers and upper case only. Character 32-94
static const unsigned char SmallFontLookup [][5] = {
    { 0x00 , 0x00 , 0x00 , 0xff , 0xff } , // sp
    { 0x17 , 0xff , 0xff , 0xff , 0xff } , // !
    { 0x03 , 0x00 , 0x03 , 0xff , 0xff } , // ”
    { 0x0a , 0x1f , 0x0a , 0x1f , 0x0a } , // #
    { 0x1f , 0x1f , 0x1f , 0xff , 0xff } , // $
    { 0x13 , 0x0b , 0x08 , 0x1a , 0x19 } , // %
    { 0x1a , 0x15 , 0x19 , 0x14 , 0xff } , // &
    { 0x03 , 0xff , 0xff , 0xff , 0xff } , // ’
    { 0x0e , 0x11 , 0xff , 0xff , 0xff } , // (
    { 0x11 , 0x0e , 0xff , 0xff , 0xff } , // )
    { 0x0a , 0x04 , 0x0a , 0xff , 0xff } , // _
    { 0x04 , 0x0e , 0x04 , 0xff , 0xff } , // +
    { 0x10 , 0x0c , 0xff , 0xff , 0xff } , // ,
    { 0x04 , 0x04 , 0x04 , 0xff , 0xff } , // -
    { 0x10 , 0xff , 0xff , 0xff , 0xff } , // .
    { 0x18 , 0x0c , 0x06 , 0x03 , 0xff } , // /
    { 0x0e , 0x11 , 0x0e , 0xff , 0xff } , // 0
    { 0x12 , 0x1f , 0x10 , 0xff , 0xff } , // 1
    { 0x19 , 0x15 , 0x12 , 0xff , 0xff } , // 2
    { 0x15 , 0x15 , 0x0e , 0xff , 0xff } , // 3
    { 0x0f , 0x1c , 0x08 , 0xff , 0xff } , // 4
    { 0x17 , 0x15 , 0x09 , 0xff , 0xff } , // 5
    { 0x0e , 0x15 , 0x09 , 0xff , 0xff } , // 6
    { 0x19 , 0x05 , 0x03 , 0xff , 0xff } , // 7
    { 0x0a , 0x15 , 0x0a , 0xff , 0xff } , // 8
    { 0x02 , 0x15 , 0x0e , 0xff , 0xff } , // 9
    { 0x0a , 0xff , 0xff , 0xff , 0xff } , // :
    { 0x10 , 0x0a , 0xff , 0xff , 0xff } , // ;
    { 0x08 , 0x0a , 0x11 , 0xff , 0xff } , // <
    { 0x0a , 0x0a , 0x0a , 0xff , 0xff } , // =
    { 0x11 , 0x0a , 0x08 , 0xff , 0xff } , // >
    { 0x01 , 0x15 , 0x02 , 0xff , 0xff } , // ?
    { 0x0a , 0x15 , 0x1d , 0x11 , 0x0e } , // @
    { 0x1e , 0x05 , 0x1e , 0xff , 0xff } , // A
    { 0x1f , 0x15 , 0x0a , 0xff , 0xff } , // B
    { 0x0e , 0x11 , 0x0a , 0xff , 0xff } , // C

    { 0x1f , 0x11 , 0x0e , 0xff , 0xff } , // D

    { 0x1f , 0x15 , 0x11 , 0xff , 0xff } , // E

    { 0x1f , 0x05 , 0x01 , 0xff , 0xff } , // F

    { 0x0e , 0x11 , 0x1d , 0xff , 0xff } , // G

    { 0x1f , 0x04 , 0x1f , 0xff , 0xff } , // H

    { 0x11 , 0x1f , 0x11 , 0xff , 0xff } , // I

    { 0x04 , 0x10 , 0x0f , 0xff , 0xff } , // J

    { 0x1f , 0x04 , 0x0a , 0x11 , 0xff } , // K

    { 0x1f , 0x10 , 0x10 , 0xff , 0xff } , // L

    { 0x1f , 0x02 , 0x04 , 0x02 , 0x1f } , // M

    { 0x1f , 0x06 , 0x0c , 0x1f , 0xff } , // N

    { 0x0e , 0x11 , 0x11 , 0x0e , 0xff } , // O

    { 0x1f , 0x05 , 0x02 , 0xff , 0xff } , // P

    { 0x0e , 0x11 , 0x19 , 0x1e , 0xff } , // Q

    { 0x1f , 0x05 , 0x1a , 0xff , 0xff } , // R

    { 0x12 , 0x15 , 0x09 , 0xff , 0xff } , // S

    { 0x01 , 0x1f , 0x01 , 0xff , 0xff } , // T

    { 0x0f , 0x10 , 0x10 , 0x0f , 0xff } , // U

    { 0x07 , 0x08 , 0x10 , 0x08 , 0x07 } , // V

    { 0x0f , 0x19 , 0x0c , 0x19 , 0x0f } , // W

    { 0x1b , 0x04 , 0x1b , 0xff , 0xff } , // X

    { 0x03 , 0x1c , 0x03 , 0xff , 0xff } , // Y

    { 0x19 , 0x15 , 0x13 , 0xff , 0xff } , // Z

    { 0x1f , 0x11 , 0xff , 0xff , 0xff } , // [

    { 0x02 , 0x05 , 0x02 , 0xff , 0xff } , // back slash
```

```
      { 0x11 , 0x1f , 0xff , 0xff , 0xff } , // ]
      { 0x02 , 0x01 , 0x02 , 0xff , 0xff }   // ^
} ;

#endif /*PCD8544_H_*/
```